UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

A Comparison Between Geometric Properties and Central Moments to Detect P300 Waves

João Ricardo Dias Cardoso

Mestrado em Informática

Dissertação orientada por:
Prof. Doutor Manuel João Caneira Monteiro da Fonseca

2018

Acknowledgments

My first acknowledgement will be to my supervisor Manuel João Caneira Monteiro da Fonseca who helped me a lot with this work which would have been impossible to accomplish without him. I recognize that he also suffered a lot with me because I am not organized and I am a terrible writer, which lead him to spend blank nights correcting my writable mistakes. I can only thank him everyday for helping me. I also want to thank my University which gave me a space (LASIGE) and Wi-Fi to conduct my work and procrastination.

I also would like to thank my group of friends: Filipe Pereira, Dharmite Prabhudas, João Rodrigues, João Batista, João Morena and José Mendes. They helped me to relieve and forget about my work, and they also helped me to procrastinate like that time when we watched the movie ”The Room”together, thank you Tommy.

I want to thank my family who were there everyday for me and allowed me to continue studying. Most of them slept terrible because I left my computer on at night conducting several tests for my Master thesis, which did a lot of noise because of the vents. Without them none of this work were possible or even me.

Lastly but not least, to my girlfriend Daniela Godinho, who I met in the last few months, who was my rock that heard and helped me overcome my problems and fears. Without her, this dissertation would have been concluded a lot latter. She also suffered like my supervisor, helping me writing this work as my personal grammar nazi.

To my FFB (Family, Friends and Baixinha).

Resumo

As Interfaces cérebro-computador (em inglês *Brain-Computer Interfaces* (BCI)) podem ser utilizadas como uma forma de comunicar sem recorrer a qualquer movimento muscular, usando apenas sinais gerados pelo cérebro e recolhidos usando eletroencefalografia (EEG). Este tipo de aplicações é adequado para pessoas com deficiências físicas pois estas não conseguem usar dispositivos como o rato ou o teclado. Um dos paradigmas das aplicações BCI é o P300, que é um sinal cerebral que acontece quando identificamos ou reconhecemos algo que estamos à espera. O P300 é um dos componentes do Event-Related potential (ERP) que representa um pico positivo localizado perto dos 300 milissegundos. Para gerar o P300 e o não P300 é usado o *oddball paradigm*, que mostra uma sequência repetitiva de estímulos compostos de *targets* e *non-targets*. Os primeiros representam o que estamos à espera.

Uma aplicação prática destes BCIs são os *Spellers* que contêm letras e que permitem aos utilizadores escreverem. Os *Spellers* acendem as letras aleatoriamente, levando assim à geração de sinais P300 quando as letras desejadas são acesas (*target*). Existem vários métodos para a deteção do P300. Contudo a maioria deles requerem treino e calibração antes da sua utilização, para atingir taxas de sucesso aceitáveis. Existem alguns que precisam de ser calibrados para cada utilizador em particular, antes de poderem ser utilizados.

Com este trabalho pretendemos desenvolver dois novos métodos para detectar o sinal P300 e compará-los para sabermos qual deles é o melhor. O primeiro método usa características físicas do sinal (forma geométrica) e o segundo método calcula os momentos centrais a partir de regiões do sinal para descrever os sinais P300 e não P300. Para ambos os métodos pretendemos que não precisem de treino ou de calibração. Para isso, estudamos as abordagens existentes para detetar o P300 e analisamos alguns *Spellers*. De entre os métodos para detetar o P300 estudamos: um que se baseiava na forma do sinal EEG, *Dynamic Time Warping*, *Linear Discriminant Analysis*, *Stepwise Linear Discriminant Analysis*, *Support Vector Machine*, *Peak Picking* e *Area Analysis*. A partir do nosso estudo, descobrimos que muitas das abordagens para detetar o sinal P300 requerem treino antes de haver qualquer tarefa de comunicação de forma a otimizar os seus resultados. Esta necessidade de treino leva a que estes classificadores tomem demasiado tempo para estarem prontos a serem utilizados. Têm no entanto o aspeto positivo de apresentarem taxas de acerto na deteção do sinal P300 elevadas. Exploramos ainda como os *Spellers*

funcionam e quais eram as interfaces existentes onde se poderia utilizar estes detetores de sinais P300. Estudamos vários *spellers* como o primeiro *speller*, que usa, uma matriz 6 por 6 com letras do alfabeto e números, o *Single Character Speller*, o *Checkerboard Speller*, o *GeoSpell* e por último um que se baseia em regiões em vez de linhas e colunas.

A nossa primeira abordagem recorre às propriedades geométricas da forma do sinal EEG. É possível descrever estes sinais com esta abordagem uma vez que os sinais P300 apresentam um pico, que é o maior pico do sinal, nos 300 milissegundos após um estímulo *target*. Por outro lado, o sinal não P300 não apresenta nenhum pico após um estímulo não *target*, apresentando apenas ondas do mesmo tamanho ao longo do tempo. Portanto, estes dois sinais apresentam diferenças geométricas que os permitem distinguir. Para a criação deste método, primeiro descrevemos todas as propriedades geométricas que tínhamos à nossa disposição, baseadas num reconhecedor de esboços, mCali [Vieira, 2014]. Adicionamos ainda novas propriedades geométricas para fortalecer a nossa hipótese de descrever os sinais. Contudo, dessas propriedades apresentadas escolhemos apenas as melhores que descrevem os sinais. No final o nosso modelo ficou constituído por 24 propriedades geométricas que melhor descrevem os sinais P300 e não P300. O nosso modelo tem o seu próprio algoritmo de pré-processamento do sinal, responsável pela escolha de um *epoch* (1000 ms), pela realização da média de todos os elétrodos utilizados para a criação de um único sinal e pela realização de outra média usando múltiplas intensificações, criando assim um sinal mais estável e fácil de classificar. No entanto, para termos um modelo o mais genérico possível, escolhemos normalizar o sinal para que o nosso modelo fosse apto a descrever e identificar sinais com diferentes amplitudes. Realizamos testes para a escolha do melhor classificador das nossas propriedades, usando vários classificadores conhecidos como *Support Vector Machine* (SVM), *Random Forest*, *AdaBoost*, *BayesNet*, *LogitBoost* e *NaiveBayes*. Com os parâmetros padrão, o SVM apresentou a melhor percentagem de deteção do sinal P300. Criamos ainda sistemas de votos para certificar que caso fossem feitas combinações com os melhores classificadores teríamos uma melhor percentagem de acerto. No entanto, os resultados obtidos não superaram os resultados do SVM. Ainda tentamos aumentar a nossa probabilidade de acerto modificando o *kernel* utilizado pelo SVM e os seus parâmetros. Usamos dois que são consideramos os melhores *kernels*: *Radius Basis Function* (RBF) e o *Normalize Poly Kernel* (NPK). No final, concluimos que o melhor classificador para o nosso modelo é o SVM com o Kernel RBF com o parâmetro Gamma a 1.

A nossa segunda abordagem é baseada em regiões do sinal, pois o sinal EEG apresenta uns componentes que precedem o sinal P300 como o N2 e o P2. Depois da escolha das regiões críticas usamos fórmulas dos momentos centrais, nomeadamente a média e o desvio padrão, para descrever essas regiões e identificar sinais P300. Escolhemos 8 regiões do sinal onde existem as diferenças necessárias para distinguir os dois sinais. De seguida, realizamos os mesmos passos utilizados na nossa primeira abordagem, tal como a escolha

das melhores características e do melhor classificador. Na escolha das melhores características concluímos que todas as 8 regiões eram as melhores para descrever os sinais. Esta abordagem também tem o mesmo pré-processamento que o primeiro modelo, isto é, o mesmo tamanho do *epoch*, a média dos elétrodos, a média de várias intensificações e por fim a normalização do sinal. No final o nosso modelo utilizou 16 características descritivas do sinal, resultantes das oito regiões e das duas fórmulas dos momentos centrais. Para a escolha do melhor classificador realizamos comparações entre dois classificadores: o SVM de *Kernel* RBF com parâmetro Gamma a 1 e o *Random Forests*. O SVM apresentou a melhor percentagem de acerto, sendo assim o classificador escolhido para o nosso segundo modelo.

Por fim, realizamos testes de comparação entre as duas abordagens para descobrir qual seria a melhor para detetar o sinal P300. Realizamos testes *user-independent*, usando os nossos dois modelos e outra abordagem de deteção, *Peak Picking*. Realizamos ainda testes utilizando diferentes *datasets*. Para isso, treinamos um modelo com um dataset e avaliamos com outro, de forma a podermos verificar se os nossos modelos conseguiam detetar P300 com sinais de amplitudes diferentes ou retirados de dispositivos diferentes. Realizamos também testes *user-dependent* usando sinais do mesmo utilizador para treinar e para avaliar. Mais uma vez comparamos os nossos modelos com o *Peak Picking*. Finalmente, comparamos os nossos modelos usando um conjunto de sinais EEG para os quais tinhamos a percentagem de acerto obtida usando o *Stepwise Linear Discriminant Analysis* (SWLDA) como classificador. Com os resultados destes testes verificamos que a melhor abordagem entre os nossos dois métodos é o que usa momentos centrais, apresentando melhores resultados nos dois primeiros testes. Contudo, o modelo geométrico teve uma probabilidade de acerto muito próxima do modelo dos momentos centrais. No último teste verificamos que o modelo dos momentos centrais mostrava os seus melhores resultados com valores em quase todos os utilizadores acima de 90%, para dois conjuntos de dados. No entanto, no dataset usado pelo SWLDA nenhum dos nossos modelos obteve uma taxa de acerto com resultados superiores a 80%. Ainda assim, o modelo que usa momentos centrais obteve resultados superiores em dois utilizadores e nos restantes utilizadores atingiu uma taxa próxima dos valores do SWLDA. Podemos concluir que o nosso modelo pode ser melhorado como trabalho futuro, acrescentando ainda mais característas ou juntando ambos os modelos num só para criar um melhor detetor do sinal P300.

Palavras-chave: Interface Cérebro-Computador, P300, Deteção de P300, Propriedades Geométricas, Momentos Centrais

Abstract

Brain-Computer Interfaces (BCI) are a way to communicate without using any muscle movement, using only signals generated by the brain and collected using Electroencephalogram (EEG). This kind of applications are appropriate for people with physical disabilities since they cannot use devices like the mouse or the keyboard. One of the paradigms of the BCI applications is the P300. This is a signal that happens when we identify or recognize something that we are waiting for. A practical application of these BCIs are the Spellers that contain letters and allow users to write. The Spellers light the letters randomly, leading to the generation of P300 signals when the desired letters are highlighted. There are several methods for detecting the P300, but most of them require training and calibration prior to use to achieve acceptable success rates. Some even need to be calibrated for each user before they can be used. With this work we intend to develop two new methods to detect the P300 signal and compare them to find the best. The first one uses physical features of the signal (geometric shape) and the second uses regions of the signals, described with central moments. For both methods we intend that they do not need individual training. To do this, we studied the existing approaches to detect P300, and analyzed some Spellers. For the creation of the first method, we described the signals using a set of geometric properties. We also conducted tests to find the best classifier and created an EEG signal pre-processing pipeline allowing our method to use signals from different record devices. For the creation of the second method we conducted the same steps, however we chose a set of regions of the signal to describe the signal and in each of these regions we used central moments to describe them. Finally, we conducted an experimental evaluation to compare our methods with others. The results showed that between our methods the best one is the central moments method, since it showed in almost all users accuracies above 90% for 2 datasets. However, the geometric models had close accuracies but not enough to overtake the central moments model. In the last dataset, from which we had the accuracy of the Stepwise Linear Discriminant Analysis (SWLDA) from the authors, none of our methods had an average accuracy value above 80%. However, the central moments model, presented results above 80% for two users and in the rest of the users presented accuracy values close to the results of the SWLDA.

Keywords: Brain-Computer Interface, P300, P300 Detection, Geometric Properties, Central Moments

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we present the motivation, the goals to fulfill, a short description of our solution to detect P300 using geometric properties and Central Moments, as well as our contributions, results and the structure of the document.

1.1 Motivation

Almost everyone can write sentences and words, as well as spell them. However, there are some people who cannot use any motor system for communication. To help these people communicating, Brain Computer Interface (BCI) Applications could be used. BCIs can be used to read brain signals recorded in electroencephalograms (EEG) and allow people to communicate without using movement, just using brain signals.

BCI applications that use P300 Signals to communicate, are mostly Spellers, consisting of a matrix of 6-by-6 with letters from the alphabet. These Spellers do not need much training and are easy to use. However, to work they use classifiers to identify the P300 Signals, which need a lot of training and calibrations, becoming a drawback for people who wants to write or communicate fast.

To classify the P300 Signals, there are several approaches using different classifiers. Typically the problem they present as a cost for their high accuracy is the need of individual training and calibration. So, we intend to create two generic models to detect P300 Signals that can be used at daily tasks quickly and effectively without requiring individual training for each user.

1.2 Goals

The goal of this work is to create two generic P300 signals identifiers. The first one uses the shape of the EEG Signal to detect and classify it as P300 target or P300 non-target using its geometric properties. The second method identifies P300 Signals using Central Moments to describe and classify regions of the signal. These classifiers should not need

individual training or calibration.

We plan to achieve this by finding the best shape features from the P300 signal using its geometric properties for the first classifier, and by peaking the best regions of the EEG Signal that helps us identify the P300 Signal with the second classifier.

Another goal is to compare our two models between them, and also with other existing approaches, to see how they behave in different contexts. In particular, we want to evaluate them in user-independent and user-dependent conditions, as well as using a dataset of signals to create the models and another dataset to test.

1.3 Developed Solution

The solution developed in the context of this work consisted in the creation of two models. One using geometric properties and the other using central moments, to describe and classify P300 Signals.

Existing classifiers used to identify P300, typically require individual training and calibration to achieve good accuracy rates.

The need of training the classifier happens because the EEG Signal varies from user to user, so the classifier needs to adapt to the user that is using the BCI. To overcome this, we created two models that do not need to be individually trained to be used. They already have samples of P300 and Non-P300 signals to help distinguish between the EEG Signals received.

Another problem is the calibration which our models do not need, being ready to be used in almost any circumstance.

1.4 Contribution and Results Obtained

At the end of this dissertation we added two new models to the vast number of classifiers that can detect P300. However, there is a very small sample of classifiers which uses geometric approaches or uses regions of the EEG Signal to identify and classify it.

Our models do not need individual training or calibration which is something that most of current classifiers need. Our models are ready to be used at any time.

With this work we were able to demonstrate that geometric properties or central moments in specified regions of the EEG signal can be used to create a model that detects P300 waves, with a good accuracy.

1.5 Structure of the document

This document is divided in five more chapters. In chapter 2 we present some work related with ours goals, namely the use of shape features of the P300 Signal, and classifiers that

are used to classify P300 signals, as well as a comparison between classifiers. Chapter 3 presents one of the solutions developed, the creation of our Geometric model, describing how to detect P300 and Non-P300 geometrically, the features used and the model settings. Chapter 4 presents the second solution, the creation of a model using Central Moments, defining what central moments are, how to detect P300 based on regions and the settings of the model. Chapter 5 presents the experimental evaluation of our two models and their comparison. Finally, in chapter 6, we present a summary of the dissertation and a conclusion of our work.

Chapter 2

Related Work

In this chapter, we present the P300 signal and some approaches, to detect it, enumerating their advantages and disadvantages.

In the second part of the chapter, we describe Spellers based on the P300 paradigm. Finally, we discuss the described works and identify their limitations.

2.1 P300 Signal

In this section, we present the P300 signal, what it is, how to generate it and why and when to use it.

2.1.1 What is?

The firsts recordings of brain activity were done by registering Electroencephalogram (EEG), the first noninvasive method of measuring brain activity for humans. From this method researchers found that electrical potential are specifically time-locked to events, and called it the event related brain potentials (ERP). ERPs are measured brain responses to specific sensory, cognitive or motor events.

Some components of the ERP use acronyms to indicate what they are. Most of them are referred by the letter N (Negative) or P (Positive), which indicate the polarity of the peak signal, followed by a number indicating the latency in milliseconds or the component ordinal position in the waveform. Some components from the ERP are: P1, N1, P2, N2, P300 (P3a and P3b), P4, N4 and P600.

P300 is one of the peaks of an ERP waveform and one of the most used components. P300 is a positive peak located 300 milliseconds after the stimulus. Figure 2.1 shows the P300 signal and some components of the ERP waveform. The P300 signal can be elicit by either visual, auditory or somatosensory.

At first, the P300 signal was connected to lie detectors because the signal created from the stimulus could not be faked. Thus, the polygraph was used to detect lies. Nowadays

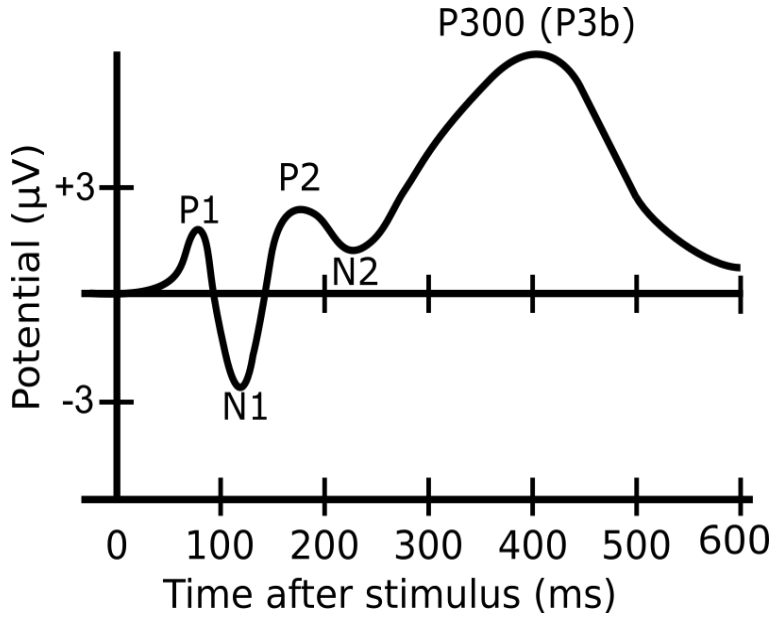


Figure 2.1: Some components from ERP, showing the location of the P300 Signal

it is used in Brain-Computer Interface (BCI) Applications, specially to provide a non-muscular communication channel, particularly for individuals with severe neuromuscular disabilities (e.g the Amyotrophic Lateral Sclerosis (ALS)).

The P300 BCI Applications are applications that use this kind of signal, to control the application. P300 Spellers are an example of an application, which uses a matrix with letters to elicit the P300 signal, by alternately lightning its rows and columns.

The P300 has two subcomponents: the novelty of P300 or P3a and the classic P300 or P3b [Polich, 2007]. The P3a is a positive-going amplitude, between 250–280 ms, that displays maximum amplitude over frontal/central electrode sites. The P3a has been associated with brain activity related to the engagement of attention, and the processing of novelty. The P3b is a positive-going amplitude that peaks at around 300 ms and correspond to the classical P300. The P3b can be generated with the oddball paradigm, or others paradigms that use the same approach, while the P3a can only be generated with three-stimulus oddball paradigm.

In the context of our work, we will focus on the classical P300.

2.1.2 How to generate?

Around 1988, Farwell and Douchin introduced the P300 BCI, in which they used the oddball paradigm to generate the P300 signal [Farwell and Donchin, 1988]. The oddball paradigm consists in showing a sequence of repetitive stimuli composed by target and non-target, and without noticing the subject's brain reacts to the target and generates a P300 signal. The P300 target is a stimulus that we want the subject to react and the P300 non-target is a stimulus to which the subject should not react to. Figure 2.2, shows an

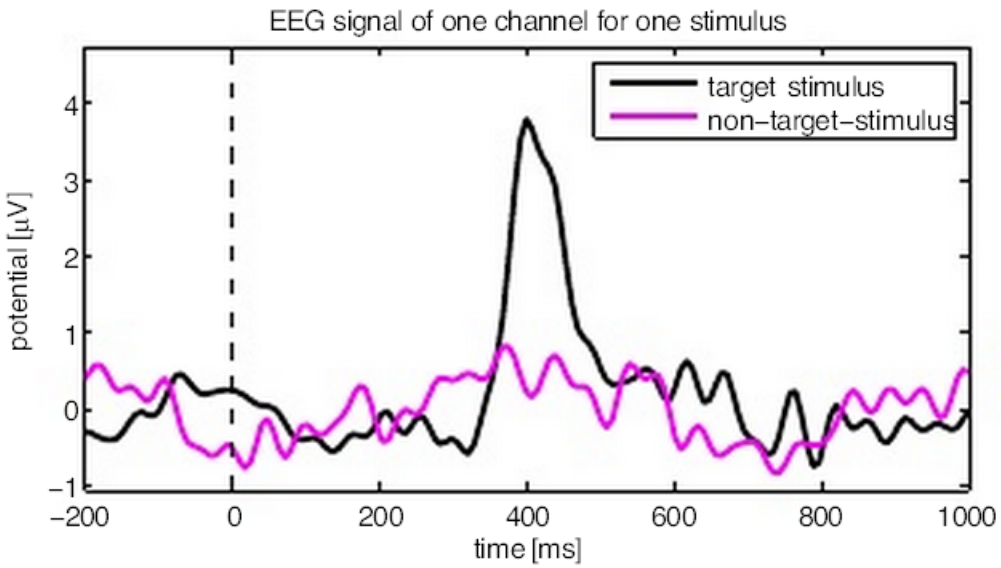


Figure 2.2: Example of an elicited P300 signal, the line in black goes up to potential 3 while not elicited P300 signal does not show that potential

example of a P300 target (black line) and P300 non-target (purple line).

The oddball paradigm is one of the most used approaches to generate P300 signals, presenting both target and non-target stimuli. However, there are other approaches, like the 'single-stimulus' paradigm [Polich and Heine, 1996] which only present a target stimulus, or the three-stimuli oddball paradigm, that is used to elicit the P3a [Wronka et al., 2008], which adds a random non-target stimulus into the mix of the target and non-target stimuli.

EEG signals are collected using several electrodes placed on the scalp according to an international standard electrode montage, called the 10-20 system (Figure 2.3).

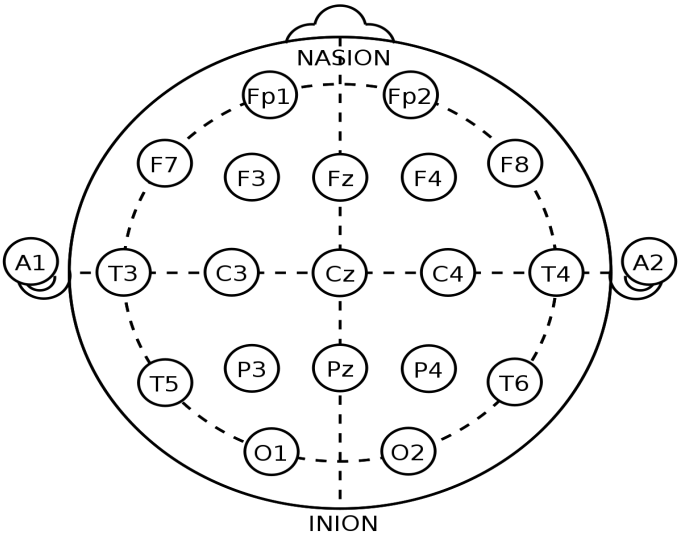


Figure 2.3: 10-20 System.

2.1.3 Why and When to use?

P300 is best used in situations that do not require intensive user training because it results from attention-based brain function. It has a wide variety of applications such as those for disabled users in home settings. The applications are relatively fast, effective and easy to use for most users.

P300 signal patterns could change in response to motivation, level of attention, fatigue, mental state and learning. Other times, users have unique EEG patterns requiring individualized calibration. To overcome these problems we need advanced digital signal processing algorithms to detect the P300 accurately and quickly.

2.2 Approaches for Detecting P300

In this section we present the main approaches used to detect the P300, focusing on the features and types of classifiers used.

2.2.1 Detection Based on EEG Shape Features

Alvarado-Gonzalez et al., used pattern recognition techniques on EGG signals to detect P300 occurrences [Alvarado-González et al., 2016]. They used a shape-feature vector, containing a contour representation based on an adapted version of the Slope Chain Code (SCC) and the tortuosity measure (a property of the contour) and used a general descriptor (the differences of areas) to describe the differences between curves. Since the SCC is expensive to compute, they adapted the SCC to create the Slope Horizontal Chain Code (SHCC). The SHCC does not compute the angle between two adjacent segments, instead it computes the slope between a segment and the horizontal in the continuous range between -90° and 90°. This way, the segments become independent, and if an electrode is disturbed it will not affect more than one chain element. Also, the SHCC does not require neither rotation invariance, since it is not designed for closed curves, nor scale invariance. To exemplify the process, Figure 2.4 shows a discretized ERP and Figure 2.5 shows the same template curve together with a non-P300 and a P300. The authors also presented an offline calibration algorithm that reduces the dimensionality of the shape-feature vector, the number of trials and the electrodes needed, and described a method to find the template that best represents, for a given electrode, the subject's P300 based on his own acquired signals.

In the experimental evaluation, authors used EEG signals from 21 students, 8 females and 13 males, between the ages of 21 to 25 years. The ten electrodes used were the Fz, C4, Cz, C3, P4, Pz, P3, PO8, Oz and PO7. They used a P300 word speller, where each row and column from the matrix was intensified 15 times every 125ms in random order, every flash lasted 62.5ms and they extracted 800ms of signal after every stimulus, for

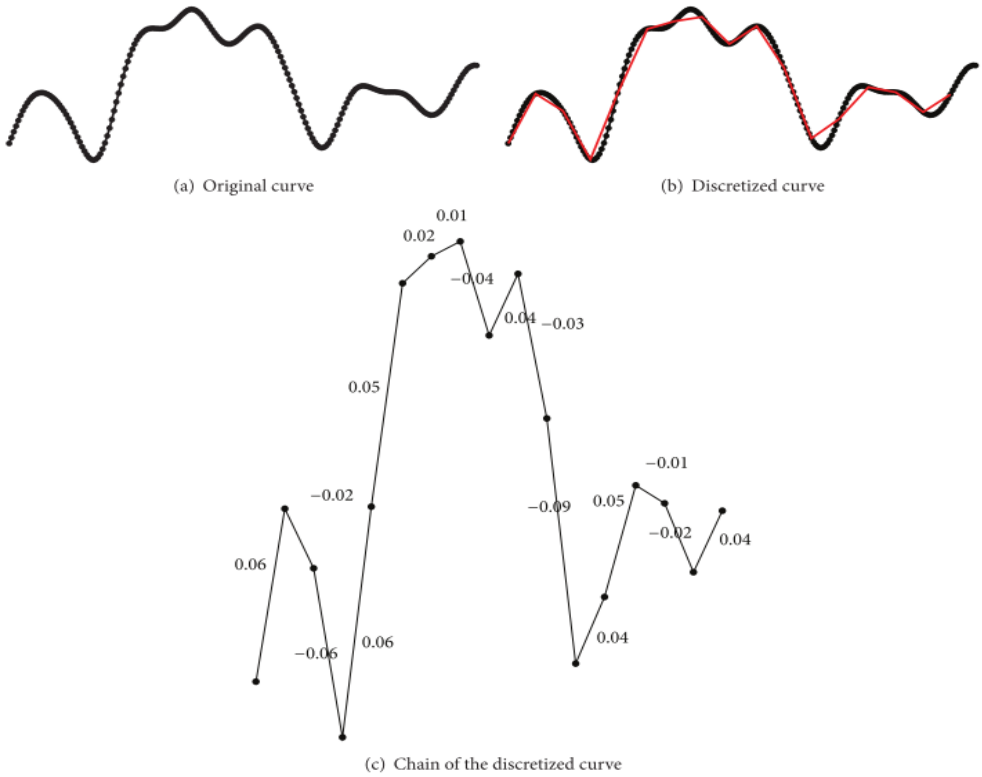


Figure 2.4: Example of a P300 discretized curve and its resulting chain code by using the SHCC method.

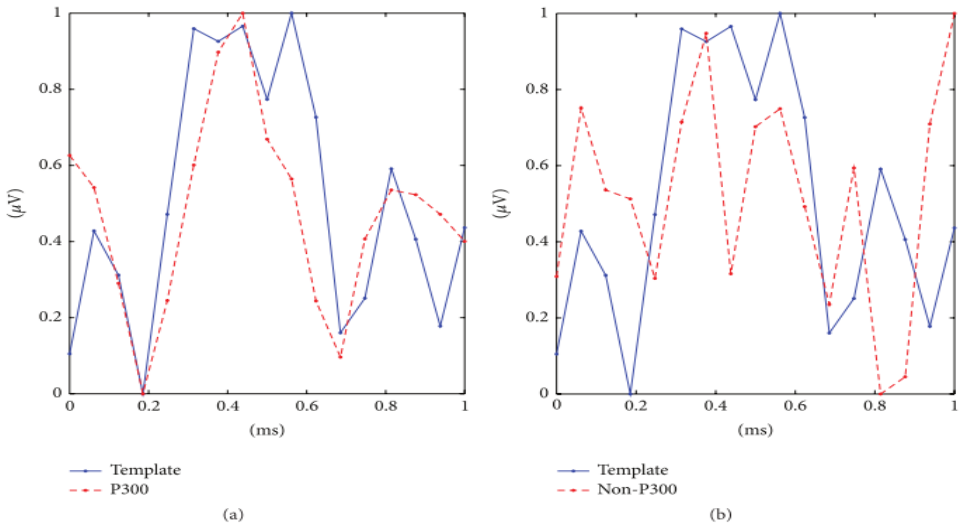


Figure 2.5: Illustration of the difference between a subject's template curve and (a) a P300 curve and (b) a non-P300 curve.

processing.

To find the best parameters for their algorithm they tested with arbitrarily chosen values, to determine the number of trials to find the template that represented the subject's

P300 signal for each electrode. After identifying the parameters, they tested the calibration algorithm using a cross-validation with a training dataset. Results showed that the P300 could be detected with less than fifteen simulations, but eight of the subjects needed no more than five stimulations. The best information was retrieved from electrodes C4, Cz, Fz, Pz, PO7, PO8 and Oz. Electrodes P3, C3 and P4 did not contribute with any relevant information.

After testing the calibration algorithm, they executed two experiments: one testing their shape-feature vector with the classifiers Stepwise Linear Discriminant Analysis (SWLDA) and Support Vector Machine (SVM), and another comparing the performance of SWLDA using the shape-feature vector versus the feature vector used by BCI2000 system.

Results from the first experiment revealed that the SWLDA showed best performance than the SVM. The recognition rate for SVM was 87% and for SWLDA was 88%. In the second experiment the percentage of correct classification with the shape-feature vector was 93.15% and with the feature vector used by BCI2000 systems was 83.18%.

In this paper, the authors detect the shape of the signal using templates with similar curves to the P300 signal. This idea requires some individual calibration and training which could be a bit of a downside. The calibration is very important and crucial but if well done assures a good detection of the P300 signal, with a high average accuracy, with less than fifteen stimulations.

2.2.2 Dynamic Time Warping

Concept

Dynamic time warping is an algorithm for measuring the similarity between two time series. It has been used in the speech domain, to cope with different speaking speeds [Berndt and Clifford, 1994]. In general, it calculates an optimal match between two given temporal sequences with certain restrictions. The DTW does not guarantee the triangle inequality to hold. Figure 2.6 shows an example of a distance measured with DTW between two signals.

Work by Casarotto

In the work of Casarotto et al [Casarotto et al., 2005], the authors developed a method based on DTW to compute reliable templates of ERP for homogeneous groups of subjects and to automatically quantify the morphological characteristics of the ERPs. As DTW compute a distance between two signals, after aligning them, they used a double-mean technique to pin-point the average value of the distance, to create the template. Figure 2.7 ilustrates the creaton of a template, from two signals, using the technique.

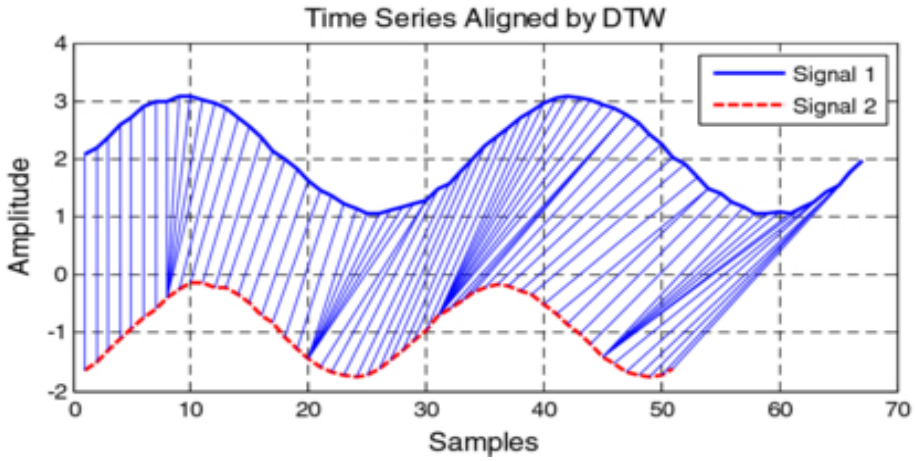


Figure 2.6: The use of the method with two signals.

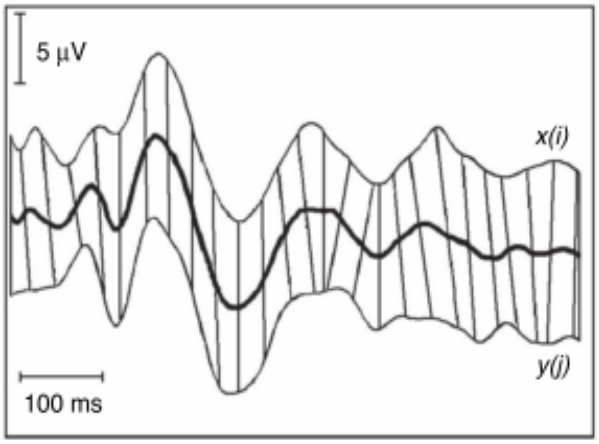


Figure 2.7: Creation of the template, using the double-mean technique.

In the experimental evaluation, they had thirty-two children with dyslexia and discrepancy. The stimuli were made using a vaccum flourescent display, presenting 21 Italian alphabetic capital and small letters. The persistence of the letters was 25ms. A minimum of four sets of stimuli were presented in the same random order for all the children. The ERPs were recorded during two conditions. The first condition, letter presentation (LPR) was passive, where the subjects passively watched letters without making any effort to read or articulate them silently. The second condition, letter recognition (LRE), was active: subjects read aloud the letters randomly appearing on the screen after the technician pressed a button.

The electrodes used were Fz, Cz, Pz, Oz, C4, C3, T4, T3, P4 and P3. Authors also captured other information like Electrooculogram (EOG), Electromyograph(EMG), Electrocardiogram (ECG) and pneumogram. To improve the signal-to-noise ratio (SNR) of averaged ERPs, a principal component analysis was applied to reduce artifacts from ocular movements and blinks. The detection was applied to the 700ms after the stimulus, allowing a reduced time of computation.

The components extracted were divided by their latency range. The components with latency less than 160ms (P0, N1, P1) belong to the prelexixal period, associated with sensory processing of stimuli. The components between 160-420ms (N2, PmaxA, PmaxB, N3) are concerned with stimulus categorization. The components after 420ms (P600a,N4 P600b) reflect long-term semantic memory and feedback processes.

The component P600b is not considered in the results because its latency in individual ERPs is often outside the upper boundary, but the rest of the components were correctly identified 68.56% of the times by the method.

The DTW proved to be useful for improving the comparison between ERPs in different subjects, because it reduces the morphological differences between signals. The method proved to successfully measure a significant percentage of the peaks and troughs present on the signal.

Work by Liang and Bougrain

In the work of Liang and Bougrain, authors tested template classifiers that use only one template, such as Point-to-Point averaging (P2P) classifier, Cross-Correlation averaging (CC) classifier and DTW. They also compared those classifiers with other methods that use multiple templates, such as Learning vector quantization (LVQ), multichannel learning vector quantization (mLVQ). To have a better comparison they used also Linear Discriminant Analysis (LDA) in the comparison [Liang and Bougrain, 2008].

They used a data set of two subjects, from the BCI competition III, based on the P300 speller. There are 85 letters for training and 100 letters for testing. For each letter, the recording consists of 15 epochs, and with each epoch, there are 12 flashings. For each epoch, a random permutation was chosen to highlight rows and columns. The matrix was 6-by-6 containing twenty-six letters, nine digits and one dash character. The electrodes used are presented in Figure 2.8.

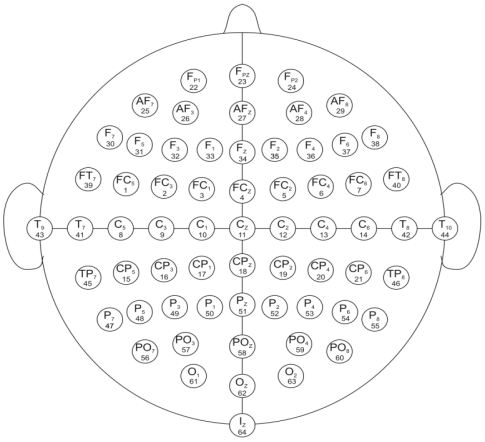


Figure 2.8: All 64 channels used in BCI competition III

In the experimental evaluation, the results were based on the raw information from the channel Cz and then by all channels from the two subjects. The results from the Cz channel showed that DTW had an accuracy of 39% and 15% for subject A and B, respectively. However, the classifier LVQ and LDA had better performance. The accuracy for LVQ was 43% and 21%, respectively subject A and B. LDA had an accuracy of 41% and 26% for the same subjects. With all channels, the results were terrible for all one template classifiers, with DTW having an accuracy of 15% and 3% for subject A and B, respectively, resulting in an average of 9% for all 64 channels. Meanwhile, mLVQ and LDA had higher accuracy. MLVQ achieved an accuracy of 87% and 96% for subject A and B, and LDA an accuracy of 88% and 96% for subject A and B.

DTW was not efficient, because producing only one ERP template, has two difficulties: first, responses are too noisy to easily distinguish ERP from non-ERP responses and second, it does not take into account the specificities of non-ERP responses to catch small differences between noisy ERP and non-ERP responses.

Work by Chakraborty and Horie

In the work of Chakraborty and Horie, they wanted to reduce the number of electrodes and the time needed to spell a letter, because the BCI spellers, on the market, use 8 electrodes and take 72 seconds to collect data [Chakraborty and Horie, 2016]. They used a cluster that contains signals with similar shape in one group, using DTW, because it would ensure that signals of similar shape, though with certain time delay, will have low distance and would be clustered in the same group. To perform clustering, they used the Ward's algorithm, a top-down agglomerative algorithm. This algorithm can set a threshold for the inter cluster distances and thus tune the number of clusters visually, using dendrogram.

In the experimental evaluation, they used a speller of 6-by-6 with 26 characters and 10 numerals. All six rows and columns flash for 10 times, randomly. The duration of the flash is 600ms, which means that for a single target character it will take 72 sec. The electrodes used were A1, A2, Fp1, Fp2, Fz, F3, F7, F4, F8, Cz, C3, T3, C4, T4, Pz, P3, T5, P4, T6, O1 and O2.

For the feature selection, they used MultiObjective Genetic Algorithm (MOGA), and for each signal they extracted two features. After that they labelled data from a subject, in signals with and without P300. They used as classifier an artificial neural network, trained using error back propagation.

Results showed that when relevant electrodes are selected for a single individual, it can reduce the number of electrodes to as low as two, and still achieved good recognition with an average accuracy of 67%. They conclude that recognition rate is higher if the subject participated in the experiment several times. Thus, subject training is required to increase concentration during the experiment, and to achieve high recognition rates.

In conclusion, DTW shows potential as classifier for one electrode only, because it creates one template only, in this way it can create a template for that electrode. However, for multiple electrodes it has some problems. It can be used in other way, like for classifying signals and joining them in cluster, but as classifier of P300 speller it needs more work. The problem the method has is the SNR, since it is hard for it to distinguish ERP from non-ERP response.

2.2.3 Linear Discriminant Analysis

Concept

The Linear Discriminant Analysis (LDA), a generalization from Fisher Linear Discriminant, [Izenman, 2008] [Kantardzic, 2011] is a feature reduction technique which is often used in machine learning and big data with the purpose to find a linear combination of features that can better separate two or more classes. These classes may be identified as species of plants, different types of tumors, etc. To distinguish the known classes from each other, it is used a unique class label. This analysis has two goals: i) discrimination, uses the information to learn and set labels to construct a classifier that will separate predefined classes, ii) classification, that given a set of non-labeled information, use the classifier to predict its class.

Work by Carabalona

Carabalona investigated the differences and performance of the P300 BCI between alphabetic and icons, and between using the colors white or green as stimulus. The author considered three factors, stimulus type, stimulus color and stimulation timing. In the case of timing it means FAST or SLOW [Carabalona, 2017]. For slow stimulation it flashes 100ms and has a 900ms of dark time between two flashes, for fast stimulation it flashes 60ms and has a dark time of 10ms. It was used data from eight subjects, collected using electrodes Fz, Cz, P3, Pz, PO7, Oz and PO8. It used two spellers, one with alphanumeric characters and another with icons. LDA was used to classify the data, after a phase of training in order to extract and learn how to classify the features.

Results showed that electrodes Pz, PO7 and PO8 had a big impact on the accuracy for the speller with characters. Without those electrodes, the results got worse. The average of accuracy for the speller using stimulus color white and timing fast, was reduced from 94% to 62% when we remove these electrodes. For the same speller using stimulus color green and timing fast the average of accuracy was reduced from 98% to 75%. With timing slow, the speller was not very affected by the removal of these electrodes achieving an average accuracy above 80%.

Work by Selim et al

Selim et al, wanted to test the performance of different machine learning algorithms for the P300 Speller paradigm based on accuracy [Selim et al., 2009]. The algorithms chosen were Bayesian Linear Discriminant Analysis (BLDA), Linear Support Vector Machine (SVM), Fisher Linear Discriminant Analysis (FLDA), Generalized Anderson's Task linear classifier (GAT) and LDA.

They used a data set of two subjects collected using all 64-channels, from the BCI competition III, based on the P300 speller. There are 85 letters for training and 100 letters for testing. For each letter, the recording consists of 15 epochs, and with each epoch, there are 12 flashings. For each epoch, a random permutation was chosen to highlight rows and columns. The matrix was 6-by-6 containing twenty-six letters, nine digits and one dash character.

Results showed that BLDA and SVM had better performance. The accuracy with 15 epochs for BLDA was 98% and SVM was 97%. Meanwhile, LDA had an accuracy of 83%.

LDA being one of the oldest classifiers is still considered one of the best for classification, because of its simplicity, being fast to classify and giving robust classification. However, it has some problems, specially if we increase the input feature spaces, it could begin to deteriorate with insufficient number of training samples. The need for training is essential for this classifier to predict the P300 signal.

2.2.4 Stepwise Linear Discriminant Analysis

Concept

Stepwise Linear Discriminant Analysis (SWLDA) is a technique used to select predictor variables that are include in a multiple regression model [Krusienski et al., 2008] [Draper and Smith, 1981]. It does a combination of forward and backward stepwise regression adding the most statistically significant predictor variable. After each new entry to the model, it does a backward stepwise regression to remove the least significant variables. This process is done until the model includes a predetermined number of terms or until no additional terms satisfy the entry criteria. SWLDA is one of the most efficient classifiers because the result happens with a non-exhaustive way. This provides an automatic feature extraction because unimportant terms are removed from the model, and thus we will have less training data to corrupt the classification result.

Work by Krusienski et al

Krusienski el al, performed a study to discover if using a larger set of electrodes, the performance of SWLDA would increase [Krusienski et al., 2008]. They used seven subjects, with the task to focus attention on a specified letter of the matrix. The rows and columns

were intensified for 100ms with 75ms between intensifications. They analyzed 800ms segments of data after the stimulus. They used a collection of channels sets illustrated in Figure 2.9.

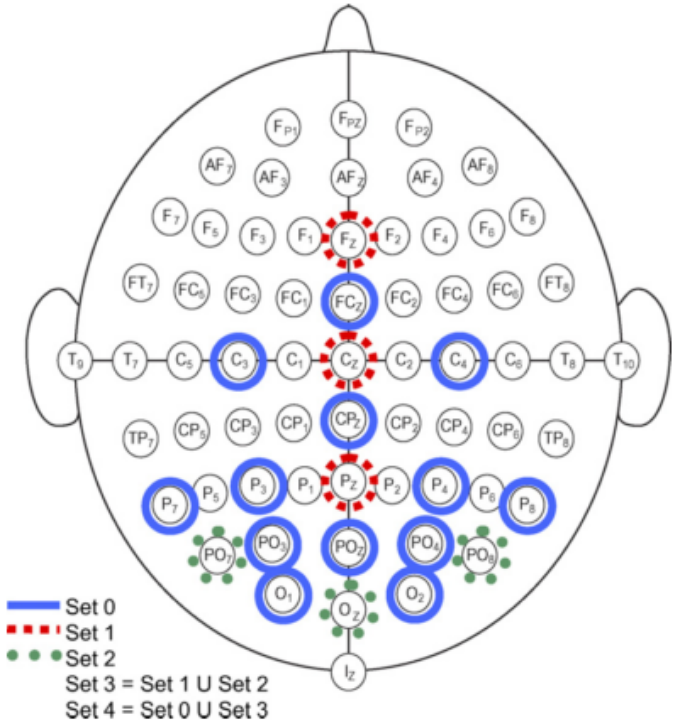


Figure 2.9: The 64-channel electrode montage and the channel sets. Set 0 is a subset defined purely for illustration purposes, sets 1–4 were used in the analysis.

Before the experimental evaluation, the classifier was calibrated to find the best features followed by four factors: channel set, reference, decimation factor and maximum features.

The results showed that the accuracy from using a larger set of electrodes to detect P300 is higher than using a small set of electrodes. The average accuracy for the larger set containing the electrodes from the Set 4 showed in Figure 2.9 was higher than 80%. Meanwhile, using the electrodes from the Set 1, the average accuracy was between 50-60%.

Work by Speier et al

Speier el al, compared an hidden Markov model (HMM), SWLDA and a naive Bayes classifier (NB) with the intend of finding which one was better [Speier et al., 2014]. HMM treats typing as a sequential process where each character selection is influenced by previous selections. The authors performed two studies, one offline and another online. For the offline study, they used ten subjects. For the online study, they used five subjects. The speller used a system of 6-by-6 matrix with a inter-stimulus interval (ISI) of 125ms. The

electrodes used were all 64-channels. The evaluation performed was based on selection rate, accuracy and information transfer rate.

Results for the offline study showed that the accuracy of the SWLDA was better than the other two classifiers with an accuracy of 88.82%. Meanwhile, NB and HMM were not far, with 88.81% and 88.34% accuracy, respectively.

The results for the online study were different with HMM having the best accuracy, with 92.34%. However, SWLDA had an accuracy of 91.68% and NB an accuracy of 82.83%.

2.2.5 Support Vector Machine

Concept

The Support Vector Machine (SVM) [Kaper et al., 2004] which is another way to do binary classification by creating a hyperplane described by the weight vector **w** and the bias term b as illustraded in Figure 2.10. This algorithm needs to acquire a hyperplane that suites a optimally criterion, by doing a projection on the weight vector. This projection would show the predicted class label. Instead of having several possible choices, it is best if it is chosen the maximum margin criterion, because it favors the hyperplane with the largest separation margin. The optimal hyperplane is best described by support vectors because those are the only vectors necessary.

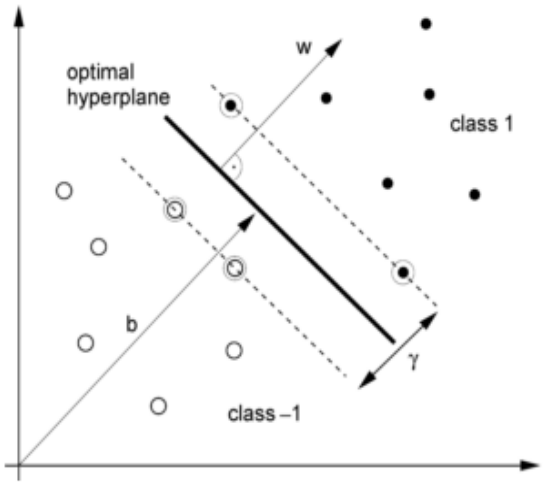


Figure 2.10: SVMs find the optimal hyperplane (solid line) to separate two classes by maximizing the margin . It can be described by the vector γ and the bias term b. Only support vectors (bordered circles) are necessary to calculate **w** and b.

Work by Kaper et al

Kaper et al, used SVM for classifying EEG signals to detect absence or presence of the P300 [Kaper et al., 2004]. They used a P300 speller paradigm represented in a 6-by-6 matrix, containing 36 symbols. Each row and column was intensified once within one trial, if the desired symbol gets intensified it is elicited a P300. The algorithm was trained with a training set labeled with ”1” and ”-1” for P300 presence/absence.

They took 600ms after stimulus from electrodes Fz, Cz, Pz, Oz, C3, C4, P3, P4, PO7 and PO8. For each trial, twelve epochs for the different rows and columns exist, in which two should contain P300 and the other ten should not.

Results using a five-fold cross-validation on the training set showed an accuracy of 84.5% for separation of P300 from non-P300 epochs. It also showed that error rates decrease with the number of repetitions from 35.5% to 0.0%, and the correct solution was found after only five repetitions.

When using the P300 speller paradigm with SVM in a very fast EEG-Based BCI, they achieved rates up to 84.7 b/min. Because the use of several electrodes could ruin the procedure, authors used only ten electrodes from the 64 electrodes. One advantage of this approach is the low preprocessing required, which is appropriate for an online solution.

Work by Mirghasemi et al

In the study of Mirghasemi et al, authors compared five classifiers to test their performance. The classifiers chosen were SVM, Gaussian Support Vector Machine (RSVM), Neural Network (NN), Fisher Linear Discriminant Analysis (FLDA) and Kernel Fisher Discriminant (KFD) [Mirghasemi et al., 2006].

In the experimental evaluation, they used the dataset from BCI 2003 competition that used P300 Speller with a matrix of 6-by-6, containing 36 symbols. All rows and columns were randomly intensified and the intensifications of each row and column were repeated for 15 times. The dataset recorded all 64 electrodes, but authors only used Fz, Cz, Pz, Oz, P3, P4, C3, C4, PO7 and PO8 electrodes.

They used Principal Component Analysis (PCA) to perform feature reduction, decreasing from 144 features to 21, and thus reducing the classification time.

Results showed perfect performance with 100% accuracy after 4 trials for several methods, namely FLDA, FKD, SVM and RSVM.

Work by Thulasidas et al

Thulasidas et al, implemented a P300 Speller using SVM as classifier and a novel feature [Thulasidas et al., 2006]. The author also performed various studies on the data to minimize the training time required, by conducting a study to minimize the number of rounds

needed for training, using an SVM model created with a subset of the training data. The results were that with two rounds for each character, the accuracy was 82.7%.

In the experimental evaluation, they used a P300 Speller with a 6-by-6 matrix of characters. All rows and columns were randomly intensified for 100ms, followed by 75 ms of no intensification. They collected data from nine subjects using 25 electrodes. The manual electrodes selected were C3, C4, Cz, CPz, and FCz, plus two distant positions P7 and P8.

Results showed an average accuracy of 95%, taking 22 seconds for each character. The time needed for training is normally 20 minutes. However, with their method of minimizing the time required for training, the training was reduced to ten minutes.

The SVM can be used as a linear classifier and as a non-linear classifier. Although it has a low processing time as classifier, it requires some time for training. In the case of a non-linear approach, we have also the problem of over-fitting, because it can model a training data very accurately but it can fail if the training data are not all representative of independent test data. SVM as another drawback that is the process of attaining suitable model and training parameters.

2.2.6 Peak Picking and Area

The Peak Picking analysis [Farwell and Donchin, 1988] is obtained by doing the difference between the lowest negative point previously from the P300 window and the highest positive point in the P300 window, as illustrated in Figure 2.11.

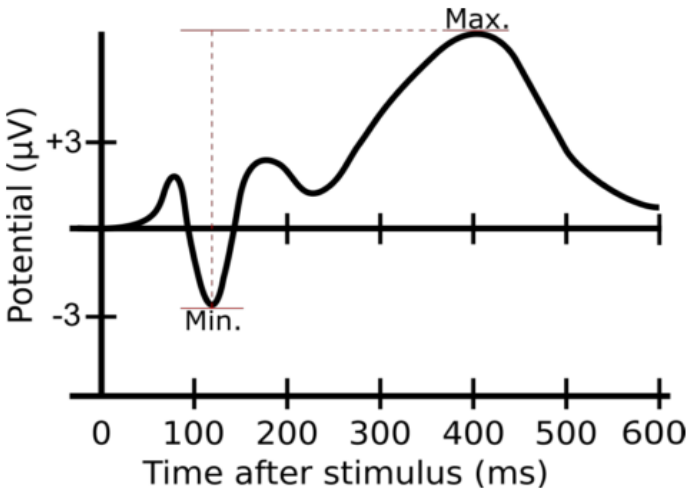


Figure 2.11: Graphical representation of the temporal evolution of a P300 EEG signal, highlighting in red the peaks used in the Peak Picking method.

Peak picking is not sensitive to latency variability, because the P300 peak can be located anywhere in a relatively wide time window. However, at a short inter-stimulus

interval (ISI) it becomes a weakness. Since, the peak picking algorithm looks for a maximum value at any point, it is susceptible to falsely attributing a P300 peak generated by a previous stimulus.

The area analysis [Farwell and Donchin, 1988], takes all the points in a broad range, in other words calculates the 'area under the curve' [Sansana, 2016] between 250ms and 600ms of a P300 from an EEG signal as presented in Figure 2.12. This way it became purely additive instead of multiplicative, and it does not require a training set. This algorithm misses some information contained in a consistent distinctive ERP shape and time course and avoids some noise. Hence, it has the advantage of information contained in a broad flat ERP.

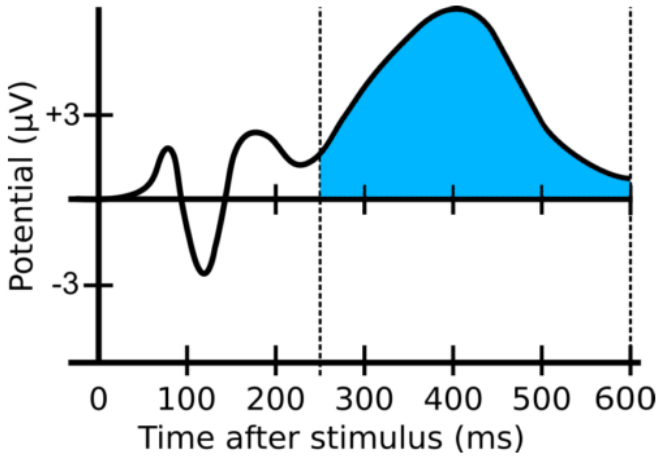


Figure 2.12: Graphical representation of the temporal evolution of a P300 EEG signal, highlighting in light blue the zone defined for the Area analysis.

In this study [Farwell and Donchin, 1988], the authors used three females and one male subjects. They used their 6-by-6 matrix containing letters of the alphabet, as well as several 1-word commands for controlling the system. Each row and columns was intensified for a period of 100msec. For the first session the ISI was 500msec, in the second session they acquired the data with both a 500msec and a 125msec ISI. The task was for the subjects to attend to a given letter and to keep a running mental count of the number of times it flashed.

The data used of the EEG was 600msec after the onset of each flash. They used four different algorithms to compute the time needed to reach an acceptable accuracy: SWLDA , Peak Picking, Area and Covariance classifiers.

Results showed that SWLDA and peak picking proved to be the most efficient algorithms. At 125 msec ISI, SWLDA was the fastest to reach 80% and 95% accuracy with average of 23 seconds and 41 seconds, respectively. At 500 msec, peak picking was the fastest to reach 80% and 95% accuracy with an average of 21 seconds and 33 seconds,

respectively. The area, took longer (50 seconds) to reach 80% and 95% accuracy.

2.3 P300 Spellers

The first speller was presented in [Farwell and Donchin, 1988], and it was a matrix of 6-by-6 containing the letters of the alphabet plus 1-word commands, as illustrated in Figure 2.13. The speller works by intensifying each of the 6 rows and columns for a period of time with an inter-stimulus interval (ISI) between intensifications. It flashes the column or row and the subject should focused on the cell (column and row) that is intensified to elicit the P300. This speller achieved a mean character selection accuracy of 85-90%.

One of the great advantages of the P300 BCI is that it does not require intensive user training, since P300 components result from endogenous attention-based brain function.

There are some issues with P300-based BCI, like the fact that the EEG signal patterns change in response to factors like motivation, level of attention, fatigue, mental state, learning, and other nonstationarities that exist in the brain. Other problems are that users may have unique EEG patterns that make it necessary for individualized calibration.

Figure 2.13: First interface of P300 Speller

One area of research has been to modify the type of visual stimulus or other stimulus to potentially elicit stronger P300, using superior flashing methods by using character motion, changing the size and sharpness from the character, attributing stimulus colors, varying the grid layout, increasing stimulus contrast or stimulus presenting ”famous faces” [Speier et al., 2017], or using varied geometric pattern stimulus [Ma and Qiu, 2017] and using icon-spellers [Carabalona, 2017].

Here we present some new paradigms for communication like the Single Character (SC) speller [Fazel-Rezai et al., 2012] that consists in randomly flashing one character at a time with a brief delay between flashes. The SC speller has a longer delay between flashes than the row/column speller (RC), allowing character classification with fewer flashes per character. The row/column flasher is about two times faster than the SC flasher. However,

the SC speller results in larger P300 amplitudes compared to the RC speller. The accuracy results for this speller reached 77.9%.

The Checkerboard Speller (CB) [Fazel-Rezai et al., 2012] [Townsend et al., 2010], was designed to correct two problems. The first is the elimination of instances when the same character flashes twice in succession. The second, is the reduction of the amount of distraction and/or inherent noise. Experimental evaluation of CB Speller, showed that it produced significant improvements in accuracy. The overall spelling accuracy for this speller was 90.6%. The CB disassociates the rows and columns of the matrix, eliminating double flashes and significantly reducing distraction, by using a matrix of 8 x 9, composed with 72 items as represented in Figure 2.14. The items in the white squares randomly populate one 6 x 6 matrix and the items in the black squares randomly populate a second 6 x 6 matrix. It is prohibited simultaneous adjacent flashes by the segregation of adjacent items into separate flash groups. The characters flashes are in sequential order: first, the rows of the white matrix, second, the rows of the black matrix, third, the columns of the white matrix and fourth, the columns of the black matrix are flashed. After all rows and columns in both matrix have been flashed, the characters in the matrix are re-randomized and the next sequence of flashes begins.

A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P
Q	R	S	T	U	V	W	X
Y	Z	Sp	1	2	3	4	5
6	7	8	9	0	.	Ret	Bs
?	,	;	\	/	+	-	Alt
Ctrl	=	Del	Home	UpAw	End	PgUp	Shift
Save	´	F2	LfAw	DnAw	RtAw	PgDn	Pause
Caps	F5	Tab	EC	Esc	email	!	Sleep

Figure 2.14: Example of the CheckerBoard Speller.

The Region based paradigm, that works by flashing several regions instead of rows or columns [Fazel-Rezai et al., 2012]. It decreases the near-target effect and human error and adjacency problem. Overall, the speller does character recognition in two levels. The first level, the characters are placed in seven groups located in different regions of the screen. Similar to the P300 BCI paradigm, to select the desired group the user attend to a specific character in one of the seven groups while each group randomly flashes. The second level, individual characters of the selected group are distributed into the seven regions. Similarly to the first level, different regions are flashed while the subject attends to one region, to select the desired character. The spelling accuracy from the results were 86.1%. Figure 2.15 is an example of the speller, at left shows the first level, on the right

shows the second level.

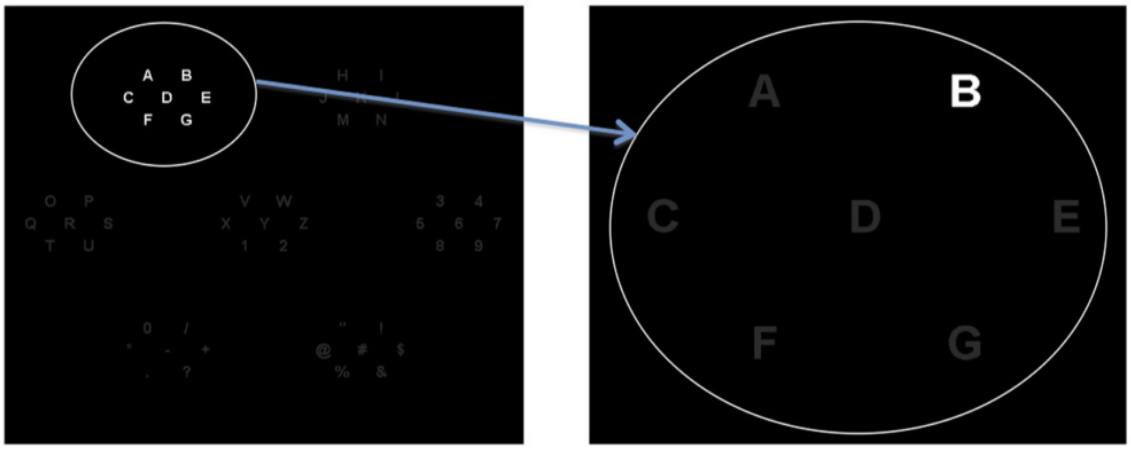


Figure 2.15: Example of the Region-based paradigm.

Aloise et al, focused a study in selective attention, where the subject can focus his attention on a specific target of the visual field *overt* and *covert* attention [Aloise et al., 2012]. Overt attention relates to the condition in which the subject turns his gaze toward the target. Covert attention, the subject focuses his attention on the target without gazing at it directly. The authors were influenced by the covert attention to develop the GeoSpell. The characters are organized with the same logic as the first speller as N by N matrix. Characters are grouped into 2N sets of N characters. With that kind of presentation, each character belongs to exactly two sets. In each set the characters are displayed at the vertices of a regular geometric figure. During the presentation, the set of characters are display sequentially on the screen and the characters are displayed at the same position for each of the two sets. All 2N sets are shown in a pseudo-random sequence that is repeated several times in a trial. A fixation point is shown in the centre to help the subject avoid eye movements.

In the experimental evaluation, the authors compared the P300 Speller and GeoSpell. The electrodes used were Fz, Cz, Pz, Oz, P3, P4, PO7 and PO8 for both Spellers. Results showed that P300 Speller had a better accuracy, with 96.17%. Meanwhile, the GeoSpell had an accuracy of 77.82%.

From the Spellers presented, the only one that showed promising results was the Checkerboard paradigm, because it was the only one that had a higher accuracy than the normal. Meanwhile, the others Spellers showed new ways to elicit the P300, but they did not reached the desired performance that would surpass its precedent.

2.4 Discussion

In this section we discuss about the performance of the classifiers described previously, to find the better classifier of P300 Signal. We also analyze the existing spellers presenting

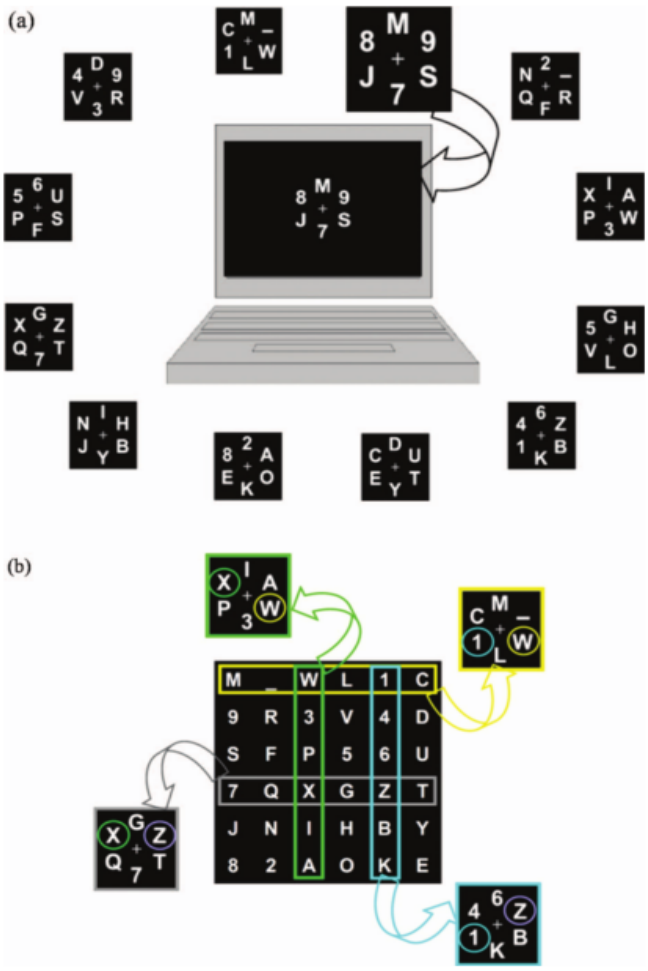


Figure 2.16: (a) Example of GeoSpell. (b) Group organization. Each group contains the characters of one row or one column of matrix.

their advantages and disadvantages.

The work of Alvarado-Gonzalez et al, is related to our goal to detect the P300 signal because it uses the shape of the signal to detect other signals with the same shape [Alvarado-González et al., 2016]. In this case, they created templates with similar curves of the P300 signal. However, for our solution the detection of P300 signal needs to be fast and should not require calibration, which is the problem they face. In their method the calibration is very important and crucial. If well done it results in a good detection of the P300 signal, with an accuracy of 88% using the classifier SWLDA.

Another algorithm that is related to our work is Dynamic Time Warping, because it can be used to create templates from P300 signals by measuring the similarity between two time series. However, it is not a very reliable classifier. When used with a single electrode the accuracy can be around 50%, while with the collection of 64 electrodes, the classifier had an average accuracy of 9%. The major problem from the method is the SNR, since it is hard for it to distinguish ERP from non-ERP response.

A classifier to be useful need to be fast during classification, require little or no training, and have a good accuracy. We presented some classifiers that are used nowadays, like the LDA, the SWLDA and SVM, and others that are not that much mention in the community like the Area and the Peak Picking.

We present in Table 2.1 the advantages and disadvantages of them. The algorithms Peak Picking and Area are not fast enough in detecting P300, they take a lot of time. The Peak Picking is good only when there is a high ISI time, and it can detect P300 much more faster than the SWLDA.

One problem found is that all of the classifiers need individual training.

SVM has a low processing time as classifier, but requires time to be trained. From the study of Thulasidas et al, it needs about 10 minutes to train and is a very complex algorithm to parameterize [Thulasidas et al., 2006]. However, the average accuracy is very high.

The LDA, being one of the oldest classifier, is still one of the best, because of its simplicity and speed to classify. However, it has problems when the input feature space increases it could begin to deteriorate with insufficient number of training set.

The SWLDA is one of the most efficient classifiers because the result happen with a non-exhaustive way and offers a solution to the problem the LDA has that is the large input feature spaces. However, it requires time for training and calibration that is crucial to the classifier.

One aspect that is important for detecting P300 is the identification of the best set of electrodes. The most used electrodes by our studies are Fz, C4, Cz, C3, P4, Pz, P3, PO8, Oz and PO7. The less used are O1, O2, FCz, CPz, P7, P8, T4 and T3. From studies between 2015-2017, the most used electrodes were Fz, Cz, Pz, P3, P4, Oz, O1, O2, C3 and C4 [Turnip et al., 2017] [Combaz and Van Hulle, 2015] [Vareka and Mautner, 2015] [Zhang et al., 2016] [Zeyl et al., 2016] [Yin et al., 2015] [Cecotti, 2015] [Akram et al., 2015]. Some also used PO7, PO8, P7 and P8 electrodes.

In the case of the spellers, there exists a lot of adaptations from the first speller, using character motion, changing the size and sharpness of the characters, using stimulus colors, varying the grid layout, increasing stimuli contrast or stimuli presenting ”famous faces”, or using varied geometric pattern stimuli and using icons-spellers.

The spellers studied that showed best accuracy with different layout than the normal speller was the Checkerboard Speller with an accuracy of 90.6%, while with the normal speller the accuracy was between 85% and 90%. One of Checkerboards's problems is that it could be very complex to develop because it uses a matrix of 8-by-9 with 72 items divide by two colors. Region-based paradigm, Single Character speller and GeoSpell are not very promising because the accuracy was smaller than the P300 Speller. For simplicity the best choice is the normal P300 speller, because it is more easy to generate and it is used a lot by the community.

Methods	Advantages	Disadvantages
LDA	Simple to calculate Provides robust classification Does not require parameterization	Large input feature spaces Requires training
SWLDA	Offer solution to large input feature spaces Non-exhaustive Automatic feature extraction Binary classification	It does not guarantees to be convergent and will not provide a model Requires training
SVM	Maximize the margin between classes Requires little training Low amount of pre-processing	Restriction of electrodes Process of suitable model and training parameters (regularization parameter and bandwidth) Algorithm complex Training is slower
Peak Picking	Not sensitive to latency variability with high ISI	Sensitive to latency with short ISI High false-positive with short ISI
Area	Purely additive Avoids noise	Lose some info of ERP shape

Table 2.1: The advantages and disadvantages from the classifiers.

2.5 Summary

In this chapter we presented what is P300 Signal, that is an ERP component representing a positive peak located 300 ms after the stimulus. We described how to generate P300 and Non-P300 signals, using the oddball paradigm by showing a sequence of repetitive stimuli composed by target and non-target. We explained the reason to use and when these kind of signals.

Then we described existing approaches for detecting P300 presenting and discussing the classifiers. The first one was detection based on EEG shape features, which uses pattern recognition techniques to detect P300. Dynamic Time Warping that computes a distance between two signals, after aligning them, and uses a double-mean technique to pin-point the average value of the distance, to create templates that are used to detect P300 signals. Linear Discriminant Analysis is a feature reduction technique used in machine learning with the purpose to find a linear combination of features that can better separate

two or more classes. Stepwise Linear Discriminant Analysis is used to select predictor variables that are included in a multiple regression model, by doing a combination of forward and backward stepwise regression adding the most statistically significant predictor variable. Support Vector Machine is a binary classifier that creates a hyperplane described by the weight vector and the bias term. Peak Picking is the difference between the lowest negative point previously from the P300 window and the highest point in the P300 window. Finally, the Area analysis that takes all the points in a broad range by calculating the area under the curve between 250 ms and 600 ms.

Lastly we described existing P300 Spellers that are used to present the stimuli that are generated by the oddball paradigm and uses classifiers to detect P300 signals. We presented the normal speller that is a matrix of 6 by 6 contaning letters of the alphabet plus 1-word command. We also presented other spellers that use different approaches like the Single Character speller, which randomly flashes one character. The Checkerboard Speller presents a speller divided by white and black items that are randomly intensified by the order: first, the rows of the white matrix, second, the rows of the black matrix, third, the columns of the white matrix and fourth, the columns of the black matrix are flashed. We then presented the region based paradigm that works by flashing several regions instead of rows or columns. Finally, we presented the GeoSpell which groups the characters into 2N sets of N character, presenting each character exactly in two sets.

Chapter 3

Geometric Detection of P300

In this chapter, we describe our approach for detecting the P300 signal using the geometric properties of the raw EEG signal. We will present the difference between P300 and Non-P300 signals geometrically, the set of potential geometric features, our approach for the pre-processing of the signal and the importance and advantages of normalizing the signal. Finally, we describe the best classifier and the set of features used in our model for P300 detection, after feature and classifier selection.

3.1 P300 and Non-P300 Geometrically

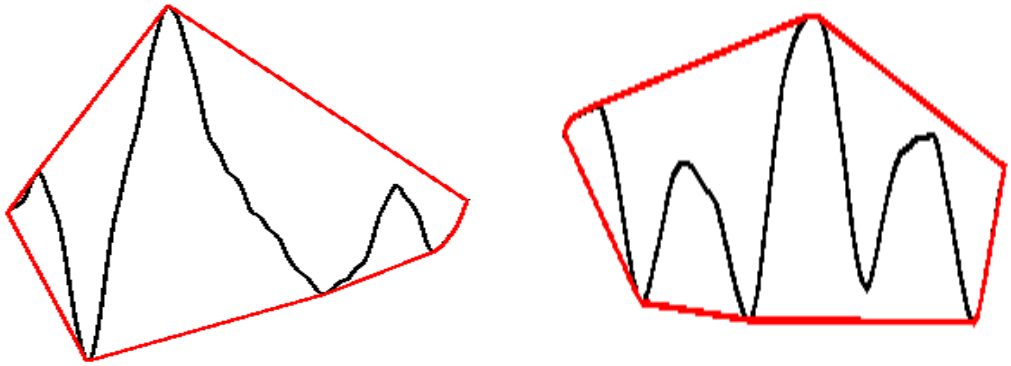
As we mentioned before, the P300 is a component from an ERP waveform, which is typically created using the oddball paradigm. A P300 Signal and a Non-P300 Signal have differences between them. The Non-P300 Signal is a signal with almost the same shape of wave along time. The P300 signal on the other hand has a peak around 300 miliseconds (ms) after the stimulus, which is the highest peak of the signal around that time. Thus, as we can see in Figure 3.1 signals have a different graphic representation.

In Figure 3.1a, we can see the representation of a P300 Signal, and in Figure 3.1b a signal that does not have a peak of P300, presenting a wave that is constant over time, representing the normal shape of a Non-P300 Signal. From those examples we can see the differences in their formats and geometry.

Therefore, we explored these particular differences between the signals, using geometric properties for describing both the P300 Signal and Non-P300 Signal. We then used those geometric features to create a model for classifying EEG signals into P300 and Non-P300 Signals.

3.2 Potential Geometric Features

In this section we describe a set of geometric features we had at our disposal from previous works related to shape recognition and new ones we thought would be beneficial to create



(a) P300 Signal (b) Non-P300 Signal

Figure 3.1: Difference between P300 and Non-P300 signal. EEG signal in black and the convex hull in red.

our P300 Detector. Most of the features presented here were used in a sketch recognizer called mCALI [Vieira, 2014], which is able to recognize multi-stroke sketches and is trainable, tolerant to different drawing orders, and invariant to the position, scale and rotation of sketches. Here, we first present the geometric features from mCALI and then we present our new features.

3.2.1 mCali Features

In this section we describe all features from mCALI.

Special Polygons

The approach used by mCALI (and its predecessor CALI [Fonseca and Jorge, 2000]) is based on a set of special polygons, from which areas and perimeters are computed, and then related among each other, to calculate ratios to be used as features. They describe the geometric properties of the sketch, or in our case, of the signal. The first step is to determine a set of special polygons and calculate the ratios between multiples properties of the polygons (area, perimeter, length, width, fill). The special polygons are Convex Hull (CH), Largest Quadrilateral (LQ), Largest Triangle (LT) and Enclosing Rectangle (ER). Figure 3.2 represents some of the special polygons referred.

For any subset of the plane (set of points, rectangle, simple polygon), its convex hull is the smallest convex set that contains that subset. The Largest Quadrilateral is the largest quadrilateral that can fit inside the convex hull. The Largest Triangle is the largest triangle that can fit inside the convex hull. The Enclosing Rectangle is the minimum area rectangle that can enclose the convex hull.

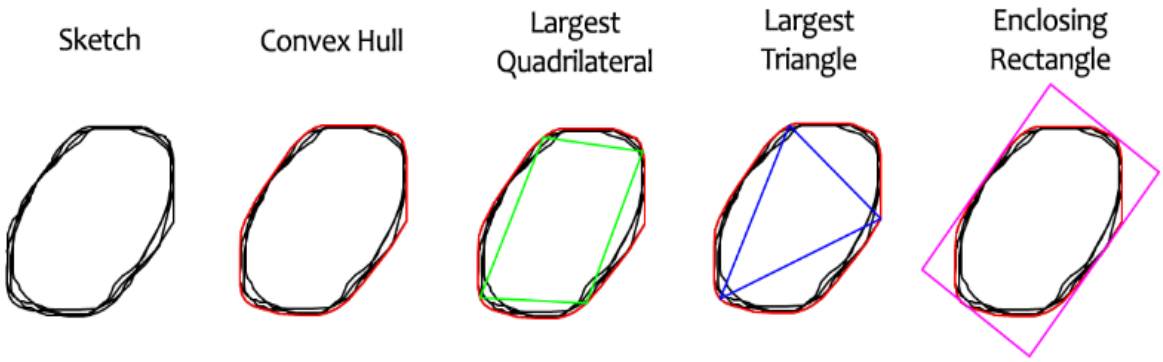


Figure 3.2: Special Polygons used in mCALI.

Ratios

From the Special Polygons it is extracted their area, perimeter, length and width to create descriptive ratios of the geometric properties. The following equations represents the ratios generated.

$$circleR = \frac{Perimeter_{CH}^2}{Area_{CH}}$$

This ratio is based on the Convex Hull and it calculates how much the sketch resemblances to a circle.

$$rectR1 = \frac{Area_{CH}}{Area_{ER}} \qquad rectR2 = \frac{Perimeter_{CH}}{Perimeter_{ER}} \qquad rectR3 = \frac{Area_{LQ}}{Area_{ER}}$$

$$rectR4 = \frac{Area_{LQ}}{Area_{CH}} \qquad rectR5 = \frac{Perimeter_{LQ}}{Perimeter_{CH}}$$

The five ratios above describe the resemblance of the sketch to a rectangle.

$$triR1 = \frac{Area_{LT}}{Area_{CH}} \qquad triR2 \ = \frac{Perimeter_{LT}}{Perimeter_{CH}} \qquad triR3 = \frac{Perimeter_{LT}^2}{Area_{LQ}}$$

The three ratios above describe the resemblance the sketch have with a triangle.

$$aspectR = \frac{Height_{ER}}{Width_{ER}} \qquad fillingR = \frac{TotalLength_{Shape}}{Perimeter_{CH}}$$

The first ratio above serves to differentiate long from more squared sketches and the last one is a fill ratio to know how full is inside the Convex Hull.

Quadrilateral of Extremes

A new poligon introduced in mCALI was the Extreme Quadrilateral (EQ), which contains vertices of four points with the biggest and lowest coordinates of the convex hull. This polygon was used to help differentiate sketches with different orientations.

$$eqR = \frac{Perimeter_{EQ}^2}{Area_{EQ}}$$

With the perimeter and area of this polygon, it was possible to create a ratio which variate with the rotation of the sketch, see Figure 3.3

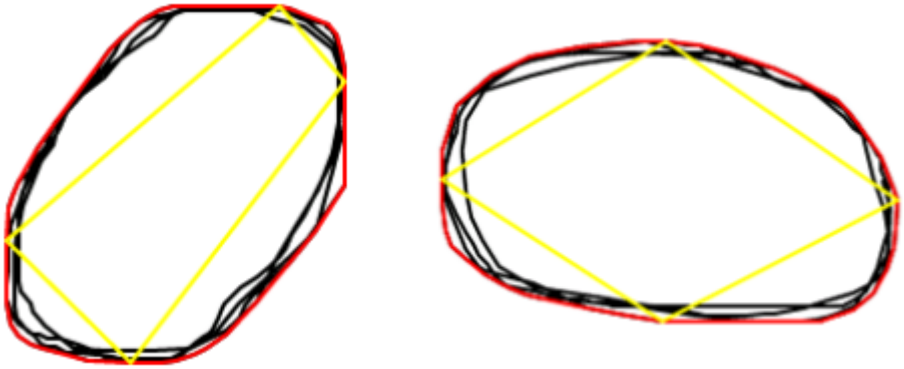


Figure 3.3: Example of Extreme Quadrilateral with the same sketch with different rotations.

Horizontal and Vertical Movement

These features obtain information about the horizontal and vertical movement of the points from the sketch. These properties are not invariant to the rotation of the sketch, helping to differentiate equal sketches with different orientations.

$$movX = \frac{x_{max} - x_{min}}{\sum_{i=1}^{N} |x_i - x_{i+1}|} \qquad movY = \frac{y_{max} - y_{min}}{\sum_{i=1}^{N} |y_i - y_{i+1}|}$$

Intersections

This property is used to distinguish between different sketches with the same Convex Hull. It creates mini polygons of the Convex Hull to detect intersections with the sketch. Later in those mini polygons are calculated new Convex Hulls to relate with the original convex hull. The process starts by calculating two polygons from the original convex hull

with scales of 0.75 and 0.5, centered in the same point, to include a big portion inside the original Convex Hull.

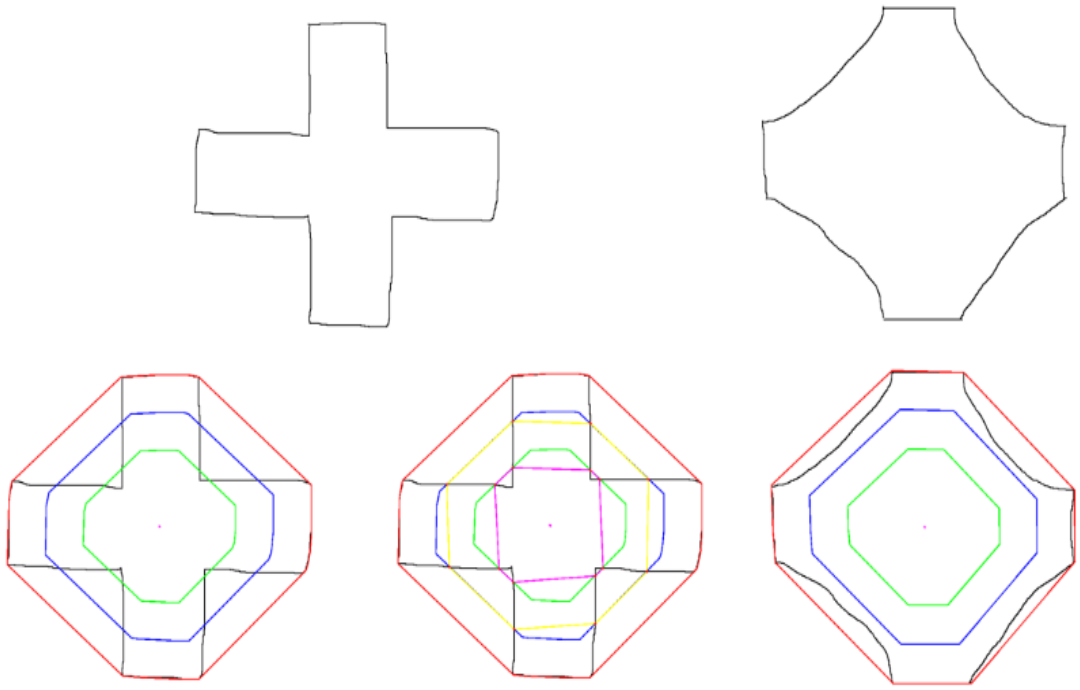


Figure 3.4: Illustration of the intersection feature, using two different sketches but with the same Convex Hull.

The first line of Figure 3.4 represents two different sketches with the same Convex Hull. In the second line, the images of the left and in the center represents the steps to generate the two new polygons for the first sketch. The image in the left demonstrate in red the original Convex Hull, in blue the polygon with scale 0.75 and in green the scale of 0.5. The point in pink represents the center. In the image in the center, the polygons in yellow and pink are the results from the intersection of the sketch with the new convex hulls presented in the left image. The image on the right shows that for the second sketch, there are no intersections with the smaller convex hulls created. Therefore, there are no intersection polygons.

From those polygons it is calculated the relations of the areas from the smaller Convex Hulls with the original, and the ratio of the area of the new two Convex Hulls resulting with the intersection of the sketch.

$$chR1 = \frac{Area_{IntersectCH1}}{Area_{CH}} \quad chR2 = \frac{Area_{IntersectCH2}}{Area_{CH}} \quad chR3 = \frac{Area_{IntersectCH2}}{Area_{IntersectCH1}}$$

Pearson's Correlation

The Pearson's Correlation allows to obtain a measure of linear dependency between two variables, with values between +1 and -1. Value 1 means that the correlation is positive, 0 means that there is no correlation and -1 means total negative correlation.

$$Correlation = \frac{COV(X,Y)}{\sigma_X \sigma_Y}$$

The σ_X and σ_Y represents the standard deviation from the coordinates.

Bounding Box

The Bounding Box is the rectangle with the lowest area that contains the sketch, in which both sides are parallel to the axis x and y. This polygon is used to create ratios between areas of the Bounding Box and the Convex Hull, and also to help differentiate rotations between sketches. Figure 3.5 represents two examples of this feature.

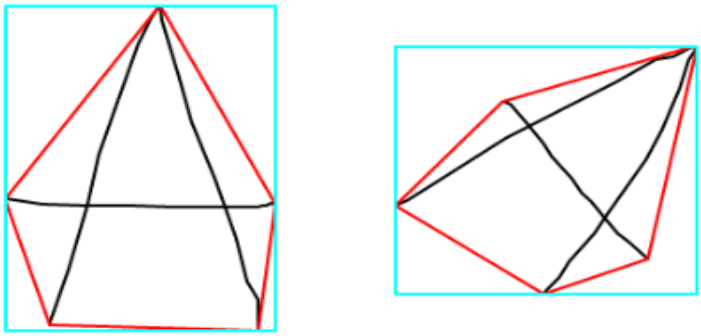


Figure 3.5: Bounding box represented in blue and convex hull in red.

$$bbchR = \frac{Area_{CH}}{Area_{BB}}$$

In the ratio above, using the bounding box and the convex hull we generated the ratio.

Quadrants

Quadrants divide the Bounding Box into four quadrants and calculate the fill ratios of each. Figure 3.6 shows what happens with the same sketch if it has different rotation, meaning that the information in each quadrants will be different.

$$quad1FillR = \frac{Length_{ShapeInQuad1}}{Perimeter_{Quad1}} \qquad quad2FillR = \frac{Length_{ShapeInQuad2}}{Perimeter_{Quad2}}$$

$$quad3FillR = \frac{Length_{ShapeInQuad3}}{Perimeter_{Quad3}} \qquad quad4FillR = \frac{Length_{ShapeInQuad4}}{Perimeter_{Quad4}}$$

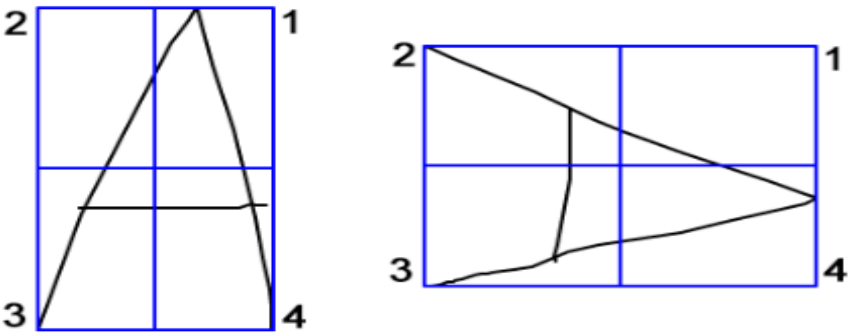


Figure 3.6: Representation of the quadrants characteristics.

3.2.2 New Features

In this section we describe the new features we introduced.

Absolute Angle Histogram

Absolute angle calculates the angle orientation of a line segment with respect to a fixed line of reference and uses the same reference for all absolute angles.

It computes an angle histogram that provides 8 intervals of values that are reduce to 4 (vertical, horizontal and the slopes) [Delaye and Anquetil, 2013].

The four features are computed as the sum of contributions from all angles to opposite directional bins. h is the histogram an n_a is the number of segments or strokes.

$$a1 = \frac{h_1 + h_5}{n_a} \qquad a2 = \frac{h_2 + h_6}{n_a} \qquad a3 = \frac{h_3 + h_7}{n_a} \qquad a4 = \frac{h_4 + h_8}{n_a}$$

Relative Angle Histogram

Another directional histogram accounts for local changes of direction. It is independent from the sketch orientation. Like for histogram of absolute orientation, the weight is calculated by the inverse of their angular distance with the central direction of the two neighboring bins [Delaye and Anquetil, 2013]. The four features are obtained from the histogram divided by n_a, which is the number of segments or strokes.

$$r1 = \frac{h_1}{n_a} \qquad r2 = \frac{h_2}{n_a} \qquad r3 = \frac{h_3}{n_a} \qquad r4 = \frac{h_4}{n_a}$$

Alpha Shape

Alpha-Shape is a concrete geometric concept that is associated with a set of points which is a generalization of the concept of the convex hull. In a way, every convex hull is an

alpha-shape but not every alpha shape is a convex hull. A real parameter, ”alpha”, controls the desired level of detail. Using the points of the Alpha Shape and the area and perimeter of the Convex Hull we created four ratios. Figure 3.7 shows the Alpha Shape and the Convex Hull from a set of points.

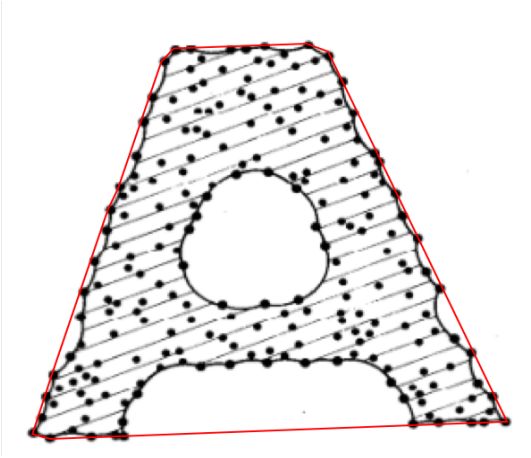


Figure 3.7: Example of a sketch. In red is the convex hull and in black is the Alpha-Shape. [Edelsbrunner et al., 1983]

$$astchAR = \frac{Area_{AS}}{Area_{CH}} \qquad astchPR = \frac{Perimeter_{CH}}{Perimeter_{AS}}$$

$$astchAB = \frac{Area_{AS}}{Area_{BB}} \qquad astchPB = \frac{Perimeter_{AS}}{Perimeter_{BB}}$$

RMS

The root mean square (RMS) is defined as the square root of the mean square. The RMS is also known as the quadratic mean and is a particular case of the generalized mean with exponent 2. RMS can also be defined for a continuously varying function in terms of an integral of the squares of the instantaneous values during a cycle. It is a way of measuring the average power of a signal.

To create our ratio we divide the RMS by the Area of the Convex Hull.

$$rms = \frac{\sqrt{\frac{\sum_{i=1}^{n} x*x}{n}}}{Area_{CH}}$$

3.3 Selection of Model Settings

Our goal is to create a P300 detector that does not need to be trained with data from the current user. Thus, we want the model to be independent from the user that is using it. To

that end, during the creation of the model, all our tests were done using the Leave-one-out method, that is, training the model with all users minus one, and then use this one for validation. However, doing this only once is not enough, so we changed the users that were used in training and validation, so that all users are used in the evaluation and used for training the model. In this section, we present the process used to develop the model. First, we describe the EEG datasets used, the pre-processing of the data, the reasons for normalizing the signal, the features selected and the classifier for the model.

3.3.1 EEG Datasets

In this subsection, we present and describe the datasets containing EEG Signals and the methods used for generating the signals.

Signal Description

An EEG dataset has a signal recorded over time by a device. This signal has a collection of intensifications or stimuli, which represents a stimulus created by the oddball paradigm, that consists in showing a sequence of repetitive stimuli composed by target and non-target. Figure 3.8, shows an example of signal description with stimuli. The stimulus is the stimulus generated by the oddball paradigm which is intensified over a short time. This stimulus may be target or non-target. After the creation of a stimulus, there is a time interval between intensifications that is called Inter Stimulus Interval (ISI). The ISI is used to separate stimulus and create time between them so that is possible to generate the other stimulus. The Stimulus Onset Asynchrony (SOA) denotes the amount of time between the start of one stimulus and the start of another stimulus.

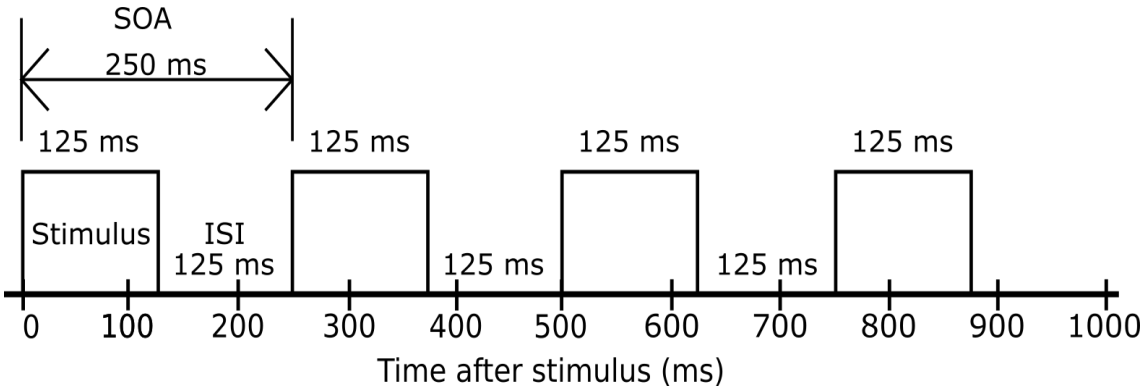


Figure 3.8: Description of signal with components of creating stimuli.

All the datasets used in our tests were collected while users used a Speller. The rows and columns were randomly intensified for 125 ms with an ISI of 125 ms which resambles the structured presented in Figure 3.8.

ALS Dataset

This dataset represents a complete record of P300 evoked potentials recorded (MOBILAB, g.tec, Austria) using the oddball paradigm. In the sessions 8 subjects with amyotrophic lateral sclerosis (ALS) [1] were used.

Each subject was presented with a 6 by 6 matrix of characters. The user's task was to focus attention on characters in a word that was prescribed by the investigator. All rows and columns were randomly intensified at a rate of 4 Hz. The EEG signals were recorded from eight channels according to 10-10 standard (Fz, Cz, Pz, Oz, P3, P4, PO7, PO8). The signal was digitized at 256 Hz and band-pass filtered between 0.1 an 30 Hz. Each row and column of the speller were intensified ten times, meaning that each character was intensified twenty times.

ERP Speller Dataset

This dataset represents a complete record of P300 evoked potentials recorded with BrainVisionAnalyzer version 2 (BrainProducts GmbH, Munich, Germany) using P300 Speller (ERP-S). In this sessions, 10 healthy subjects were used to conduct the experiment [2]. The interface was organized in a 6 by 6 matrix. The stimulation consisted in intensification of the rows and columns.

ERP GeoSpeller Dataset

This dataset represents a complete record of P300 evoked potentials recorded with g.USBamp amplifier (g.Tec, Austria), using the GeoSpell interface (GEO). In this sessions, 10 healthy subjects were used to conduct the experiment [3].

As explained before in chapter 2.3 , the characters are organized with the same logic as the first speller as N by N matrix. Characters are grouped into 2N sets of N characters. With that kind of presentation, each character belongs to exactly two sets. In each set, the characters are displayed at the vertices of a regular geometric figure, presenting 6 characters at a time. New sets of 56 characters are presented in a sequence, until all 36 have been delivered twice after 12 intensifications.

For both datasets (ERP-S and ERP-G) each subject attended three recording sessions. The EEG signals were measured from the electrodes Fz, FCz, Cz, CPz, Pz, Oz, F3, F4, C3, C4, CP3, CP4, P3, P4, PO7 and PO8. The signal was recorded using an amplifier and digitized at 256 Hz, high pass and low pass-filtered with cutoff frequencies of 0.1 Hz and 20 Hz. The trial consisted of eight stimulation sequences, and thus, sixteen intensifications of the target character. Because these two datasets have more electrodes than the

[1] https://lampx.tugraz.at/ bci/database/008-2014/description.pdf
[2] https://lampx.tugraz.at/ bci/database/009-2014/description.pdf
[3] https://lampx.tugraz.at/ bci/database/009-2014/description.pdf

ALS dataset, for our work we only used the same eight electrodes that are available in the ALS dataset (Fz, Cz, Pz, Oz, P3, P4, PO7, PO8). The reason behind this is because we wanted to create the same experiment using all datasets with the same metrics and process, having a fair comparison between them.

3.3.2 EEG Signal Pre-Processing

The pre-processing is one of the most important parts of detection, because it will define the quality of the signal to be classified. We defined three steps to accomplish our goals: epoch size, average between electrodes and average between intensifications (Figure 3.9).

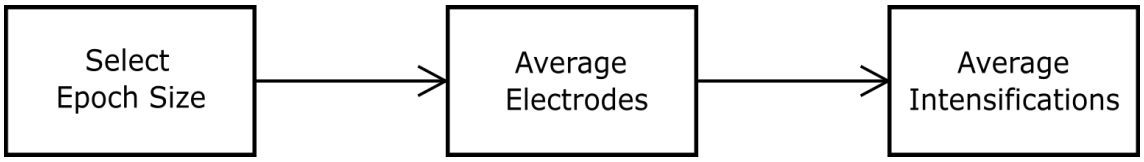


Figure 3.9: Steps of signal pre-processing.

The first step is the definition of the epoch. From existing works we have seen that the size of the epoch is normally between 700 ms and 800 ms after the stimulus, because it can acquire the P300 that is located around the 300 ms after the stimulus. However, in some cases, the P300 does not happen exactly around the 300 ms. In cases of fatigue or slow reactions the P300 can be localized after the 300 ms, in some cases around 500 ms and 800 ms, and that is the main reason for some researchers acquiring 700 ms or 800 ms of epoch. In our case, we selected 1000 ms of the signals because with that size the shapes of the P300 and non-P300 signals are more distinct geometrically.

The second step of the pre-processing is to create a single signal based on all electrodes used. This step is used by several researchers because having multiple signals to work on is hard, and by creating an average signal with all electrodes is possible to reduce the noise from the signal and create a better signal to analyze. So, we took 1000 ms from each electrode after the stimulus and computed an average signal with signals from all the electrodes.

The final step was the average of several intensifications. With this, we get a signal that is more stable and has better quality. So, after creating an average signal with all electrodes we create a new signal by averaging multiple intensifications.

3.3.3 Normalization of the EEG Signal

One question we had in our mind was if it was needed to normalize the signal or if we could keep it in its natural form, because we wanted to create a classifier to work with any recorded signal. The problem was that signals had different amplitudes and it could jeopardize our results and the geometric approach we were trying to use.

So, we performed a test with normalized and non-normalized signals to see if there were any differences between them. For this, we used two of the most common classifiers (Support Vector Machine and Random Forest) and all the features explained in Section 3.2. For the normalization of the signal, we divided each value of the signal by the absolute maximum from the collection of values inside the signal. This creates a signal with values between -1 and +1. Before using the signal to train the model and to classify, we multiplied the values of x, that represents the 1000 ms, by 2 and the values of y by 150, to amplify the signal so the geometric properties could be computed correctly. The results from our tests using normalized and non-normalized signals are presented in Tables 3.1 and 3.2. Additionally, we also studied the effect of the numbers of intensifications in the results.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
ALS Non-Nor Random Forest	65.6	64.6	66.0	66.5	70.4	71.8
ALS Non-Nor SVM	50.8	50.0	68.0	**69.7**	70.1	56.3
ALS Nor Random Forest	**68.9**	67.4	**69.3**	69.5	**72.9**	**72.8**
ALS Nor SVM	51.4	**68.1**	67.1	53.1	70.6	**72.8**

Table 3.1: Accuracy results from Non-Normalization and Normalization with ALS dataset for 5 intensifications to 10 intensifications, using SVM and Random Forest.

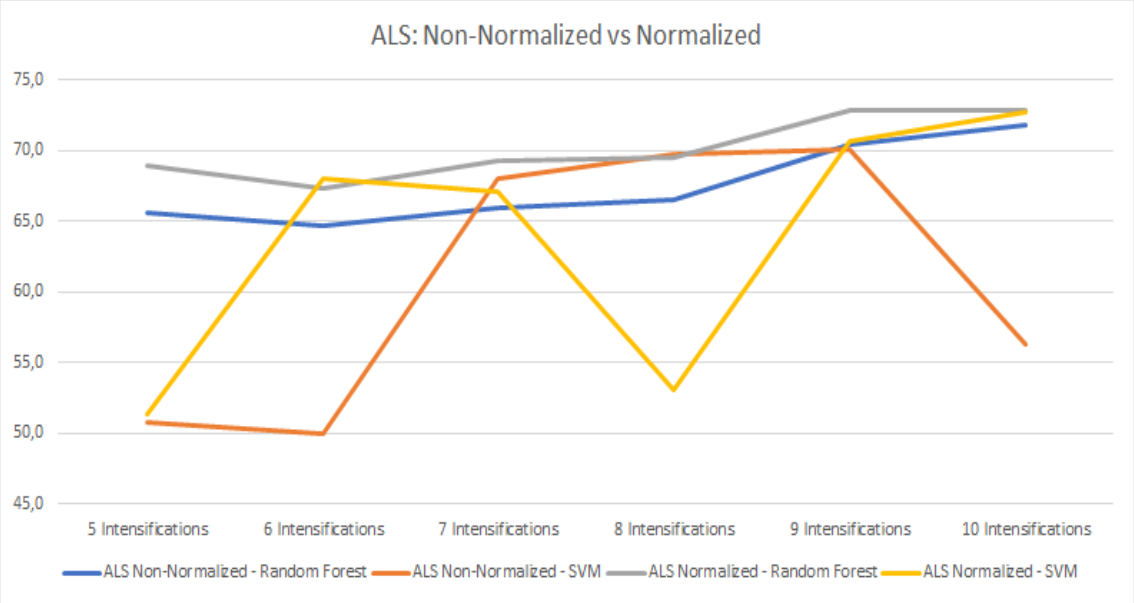


Figure 3.10: Chart of the results from the Table 3.1, using ALS dataset. Y is the accuracy and X the number of intensifications.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
ERP-S Non-Nor Random Forest	**78.2**	**77.6**	**79.1**	78.3	**79.3**	**83.9**
ERP-S Non-Nor SVM	52.5	51.5	51.9	**80.2**	78.7	80.9
ERP-S Nor Random Forest	74.2	74.4	76.0	78.1	77.8	81.8
ERP-S Nor SVM	52.4	51.9	51.7	52.9	76.3	81.6

Table 3.2: Accuracy results from Non-Normalization and Normalization with ERP-S dataset for 5 intensifications to 10 intensifications, using SVM and Random Forest.

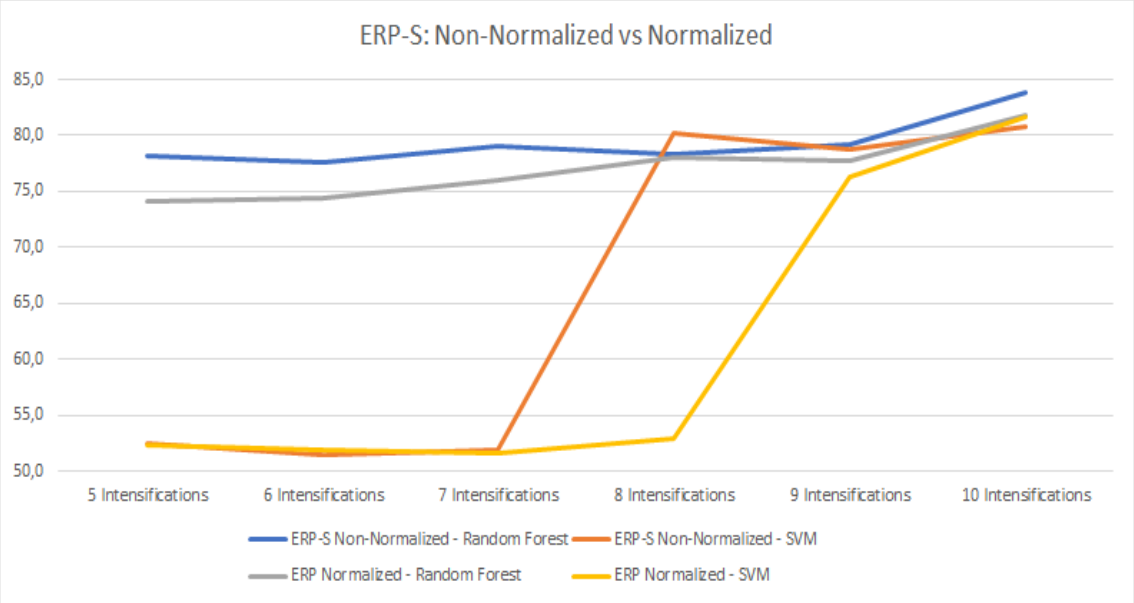


Figure 3.11: Chart of the results from the Table 3.2, using ERP-S dataset. Y is the accuracy and X the number of intensifications.

For ALS dataset the best results were with normalized signals using both classifiers Random Forest and SVM, 72.8% of accuracy with 10 intensifications. The non-normalized signals the results do not reach the results of the normalized signal, with 71.8% for Random Forest with 10 intensifications, although it is close to the normalized signal. However, with SVM it had an accuracy of 56.3%. Figure 3.10 presents the progress of the classifiers through intensifications. We can see that the SVM classifiers shows some instability, having results reaching 70.1% and then lowering to 56.3% for non-normalized signal using SVM for 9 intensifications and 10 intensifications, respectively. It happens the same problem for the normalized signal for 7 intensifications and 8 intensifications, with 67.1% of accuracy and then lowering to 53.1%, respectively.

For the ERP-S dataset the best result were for non-normalized signal, using Random Forest, with an accuracy of 83.9% for 10 intensifications. Comparing both non-normalized and normalized signals, using Random Forest, both presents stability and a satisfying accuracy. However, the non-normalized showed better results with 83.9% of accuracy, while the normalized had an accuracy of 81.6%, both for 10 intensifications.

In Figure 3.11, we can find the same situation we had using the ALS dataset, which is the instability of the SVM classifier. The oscillation of the accuracy is a problem for this model.

From these tests we could verify that using normalized or non-normalized data would not affect much the results. However, if we train the classifier with a dataset (e.g ALS) and test with another (e.g ERP-S) using non-normalized signals, the results would be very different because the ERP-S EEG signals have higher amplitudes than the ALS dataset, producing very different signals which will generate different features.

Based on these results, we decided to normalize the signal so that our model can classify using different types of signals with different amplitudes. In the end, we added another step to our EEG pre-processing signal which is the signal normalization. Figure 3.12 represents all our steps of EGG signal pre-processing.

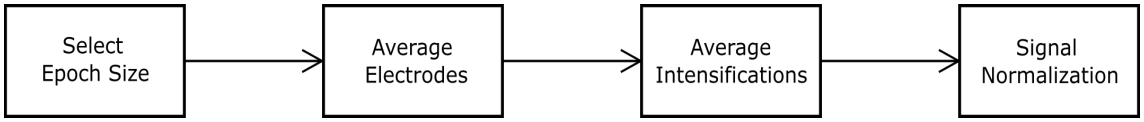


Figure 3.12: Steps of EEG Signal Pre-Processing.

3.3.4 Feature Selection

With so many features at our disposal, we investigated if all of them were needed, and what subset would produce the best results. To that end we created models using ALS and ERP-S datasets, separately. We created models using different numbers of intensifications, which went from 5 to 10. Our goal was to see if the number of intensifications and the type of dataset influenced the set of features that produced the best results.

Using the application Weka [Witten et al., 2016], we did a feature selection using GreedStepwise as search method and CfsSubsetEval as attribute evaluator for all features computed. We also used the Ranker search method with the attribute evaluator CorrelationAttibuteEval to validate our results. From the results of this feature selection we chose the best set of features to create our model and used the Ranked method to add other features for complementing our model.

Table 3.3 shows the results for each dataset (ALS and ERP-S) and Table 3.4 shows the final 24 features that were selected after the feature selection. From the ALS and ERP-S results we did a reunion of the results which resulted in 20 features. Then from using Ranker we decided to add the others absolute and relative angle histogram (a1, a4 e r1). We also found that fillingR would benefit our model, since it presented good results in classifing the signals. By adding these 4 features to the set, we created our 24 features.

ALS Dataset	ERP-S Dataset
astchAB	astchAR
astchAR	astchPR
astchPR	bbchR
bbchR	chR2
chR2	chR3
chR3	a2
eqR	a3
r4	r2
MovY	r3
rectR2	MovY
rectR3	quad2FillR
	quad3FillR
	quad4FillR
	rectR1
	rectR2
	RMS

Table 3.3: Results from the feature selection to ALS and ERP-S datasets.

Name	Formula	Description
astchAB	$\frac{Area_{AS}}{Area_{BB}}$	Relations between the alpha-shape and the convex hull and bounding box
astchAR	$\frac{Area_{AS}}{Area_{CH}}$	
astchPR	$\frac{Perimeter_{CH}}{Perimeter_{AS}}$	
bbchR	$\frac{Area_{CH}}{Area_{BB}}$	Relation between areas of Convex Hull and Bounding Box
chR2	$\frac{Area_{IntersectCH2}}{Area_{CH}}$	Relation between intersections Convex Hull with Convex Hull of sketch
chR3	$\frac{Area_{IntersectCH2}}{Area_{IntersectCH1}}$	Relation between the intersections Convex Hulls
fillingR	$\frac{TotalLength_{Shape}}{Perimeter_{CH}}$	Fill of sketch inside the Convex Hull
eqR	$\frac{Perimeter^2_{EQ}}{Area_{EQ}}$	Relation between area and perimeter from Extreme Quadrant
a1	$\frac{h_1+h_5}{n_a}$	Absolute angle histogram for segments of orientation
a2	$\frac{h_2+h_6}{n_a}$	
a3	$\frac{h_3+h_7}{n_a}$	
a4	$\frac{h_4+h_8}{n_a}$	
r1	$\frac{h_1}{n_a}$	Relative angle histogram for local changes of direction
r2	$\frac{h_2}{n_a}$	
r3	$\frac{h_3}{n_a}$	
r4	$\frac{h_4}{n_a}$	
MovY	$\frac{y_{max}-y_{min}}{\sum_{i=1}^{N}\lvert y_i-y_{i+1}\rvert}$	Vertical displacement of points
quad2FillR	$\frac{Length_{ShapeInQuad2}}{Perimeter_{Quad2}}$	Fill of the sketch inside the quadrants
quad3FillR	$\frac{Length_{ShapeInQuad3}}{Perimeter_{Quad3}}$	
quad4FillR	$\frac{Length_{ShapeInQuad4}}{Perimeter_{Quad4}}$	
rectR1	$\frac{Area_{CH}}{Area_{ER}}$	Features of similarity with a rectangle
rectR2	$\frac{Perimeter_{CH}}{Perimeter_{ER}}$	
rectR3	$\frac{Area_{LQ}}{Area_{ER}}$	
rms	$\frac{\sqrt{\frac{\sum_{i=1}^{n} x*x}{n}}}{Area_{CH}}$	Generalization of the mean integral of the squares

Table 3.4: Final set of features, resulting from the feature selection process, using the ALS and ERP-S datasets.

3.3.5 Classifier Selection

With the help of the Weka software, we used its Auto-Weka feature, to identify the classifier that produced the best accuracy. With this application we learn a lot about our dataset and our features. One of the best classifiers identified was the Random Forest, followed by others like AdaBoost, Bagging, BayesNet, LogitBoost, NaiveBayes and SVM.

With this collection of classifiers, our next step was to identify the best classifier for

our model, using the features selected. To that end, we used the ”leave one out” method to train and evaluate. We conducted multiple tests with all the classifiers enumerated before. The classifiers were implemented using the default parameters from Weka. The results for each of the classifiers are presented in Tables 3.5 and 3.6. Figures 3.13 and 3.14 represent the behavior of the classifiers and their evolution for the different number of intensifications.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
AdaBoost	**70.6**	63.8	65.2	71.0	72.1	71.0
Bagging	68.9	68.4	67.8	70.0	71.8	71.9
BayesNet	63.4	64.6	62.4	65.4	66.7	67.6
LogitBoost	69.8	65.5	66.0	71.1	71.8	70.0
NaiveBayes	63.3	64.5	61.4	63.4	64.8	67.6
Random Forest	67.9	67.7	69.4	70.1	71.8	72.1
SVM	69.1	**69.1**	**69.8**	**72.3**	**72.6**	**73.6**

Table 3.5: Accuracy results for all chosen classifiers with ALS dataset from 5 intensifications to 10 intensifications.

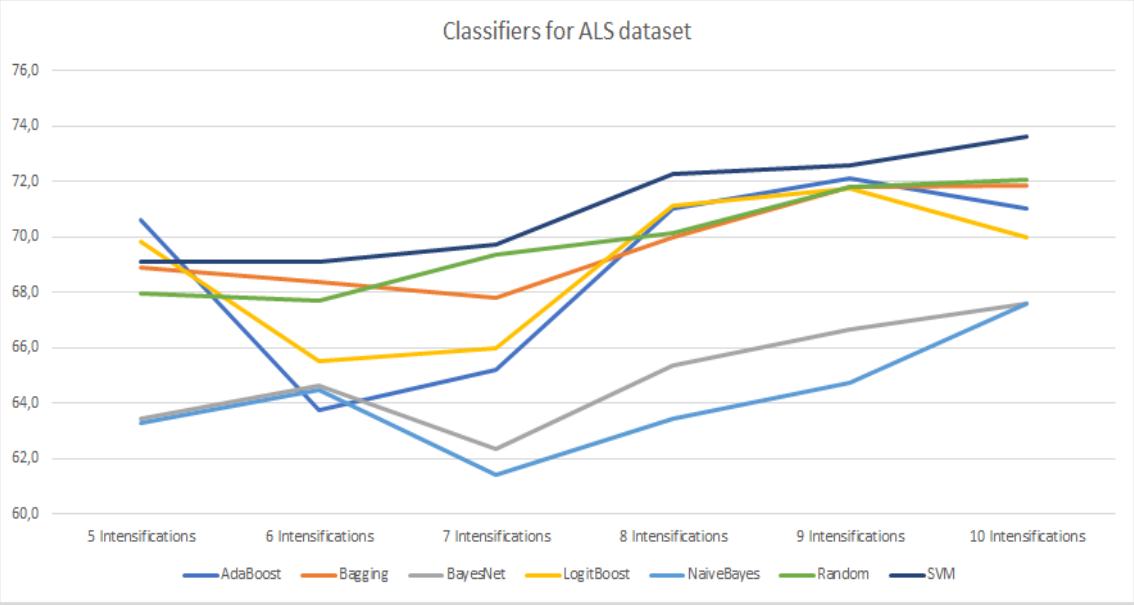


Figure 3.13: Chart of the results from the Table 3.5, using ALS dataset. Y is the accuracy and X the number of intensifications.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
AdaBoost	71.6	72.5	75.2	75.7	75.9	81.6
Bagging	**73.8**	73.2	75.0	79.3	78.7	81.6
BayesNet	71.2	70.3	71.6	74.5	74.7	78.7
LogitBoost	71.5	72.7	76.0	77.4	77.8	82.0
NaiveBayes	70.0	68.5	72.7	75.1	74.1	77.2
Random Forest	72.8	74.2	76.6	77.6	**78.9**	82.8
SVM	73.1	**77.6**	**77.1**	**81.1**	78.6	**84.1**

Table 3.6: Accuracy results from all chosen classifiers with ERP-S dataset from 5 intensifications to 10 intensifications.

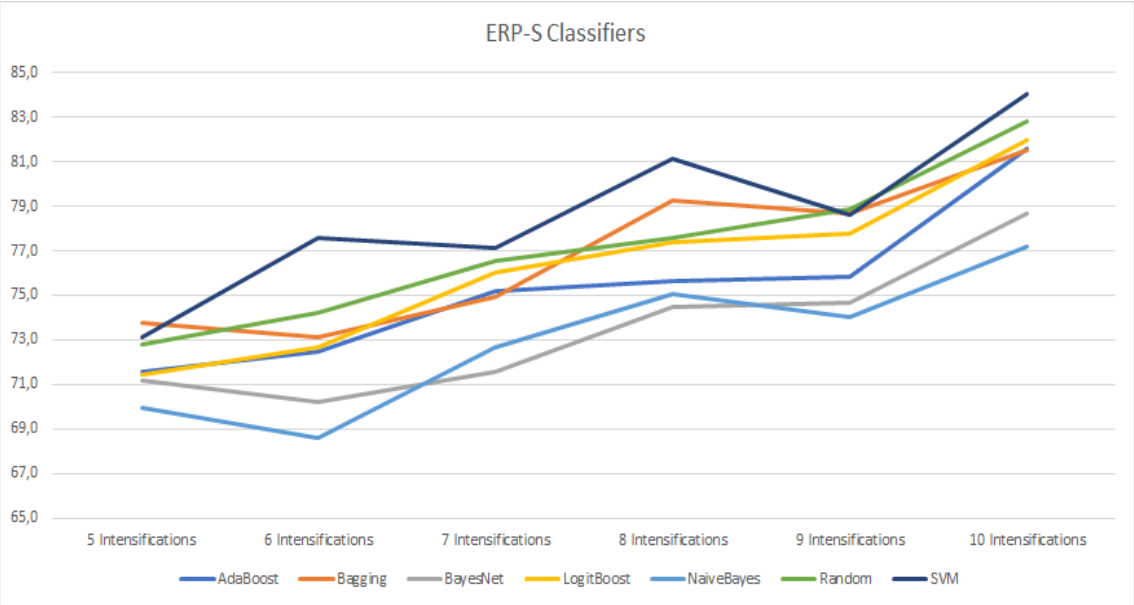


Figure 3.14: Chart of the results from the Table 3.6, using ERP-S dataset. Y is the accuracy and X the number of intensifications.

The best classifier in our tests was the SVM, achieving 73.6% and 84.1% for 10 intensifications for the ALS and ERP-S datasets, respectively. We found a bit strange that the SVM was the best classifier, because in the tests with normalized and non-normalized signals the results was backwards with Random Forest being the best classifier. The the most probable reason for this change can be the lower number of features, which improved the classifier and made it more stable.

The other classifiers, in some cases were close to the SVM, like for instance the Random Forest with 72.1% and 82.9% using 10 intensifications for ALS and ERP-S, which was the second best classifier. The other classifiers were AdaBoost, Bagging and LogitBoost.

We created vote systems using the best classifiers identified to see if we could improve the results. To that end, we created 5 vote systems that are described in Table 3.7 and their

results are presented in Tables 3.8 and 3.9, together with the results from the best classifier identified before (SVM).

Vote Name	Combinations of classifiers
V1	SVM + Random Forest + Bagging
V2	SVM + Bagging + AdaBoost
V3	SVM + Random Forest + AdaBoost
V4	Random Forest + Bagging + AdaBoost
V5	SVM + Random Forest + LogitBoost

Table 3.7: Vote Systems created, and their composition.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
V1	68.9	**69.1**	**70.6**	71.6	72.3	**73.6**
V2	**70.9**	68.1	69.3	72.1	71.9	73.1
V3	70.1	67.6	70.4	**72.3**	71.8	73.0
V4	69.7	67.5	69.8	70.9	72.3	72.1
V5	69.9	68.1	70.2	**72.3**	71.6	72.3
SVM	69.1	**69.1**	69.8	**72.3**	**72.6**	**73.6**

Table 3.8: Accuracy results from all votes and from SVM using ALS dataset, from 5 intensifications to 10 intensifications.

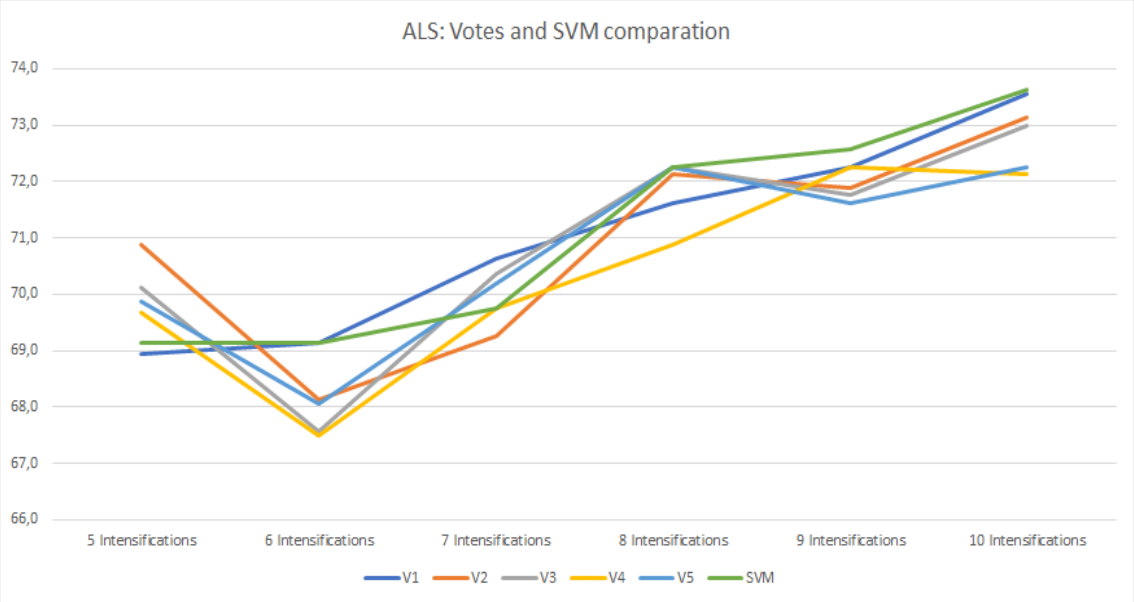


Figure 3.15: Chart of the results from the Table 3.8, using ALS dataset. Y is the accuracy and X the number of intensifications.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
V1	74.1	75.6	77.2	79.6	**79.9**	83.4
V2	73.5	74.7	77.3	79.2	78.9	84.0
V3	**74.3**	74.9	77.3	79.2	78.0	83.8
V4	73.7	74.2	**77.4**	79.2	79.2	82.5
V5	73.9	75.5	77.0	78.9	78.8	83.3
SVM	73.1	**77.6**	77.1	**81.1**	78.6	**84.1**

Table 3.9: Accuracy results from all votes and from SVM using ERP-S dataset, from 5 intensifications to 10 intensifications.

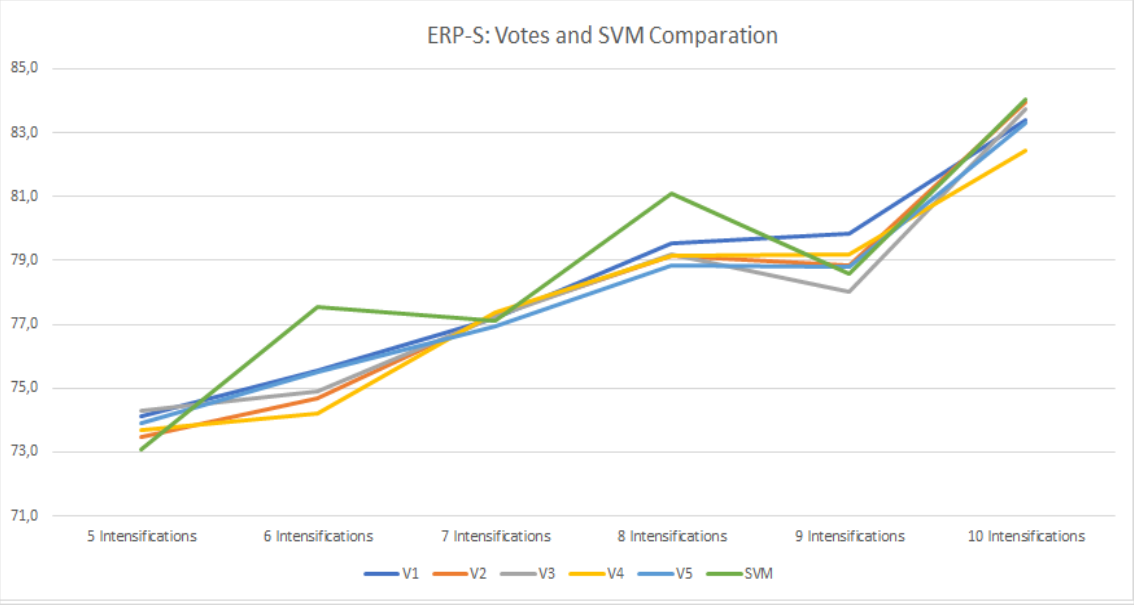


Figure 3.16: Chart of the results from the Table 3.9, using ERP-S dataset. Y is the accuracy and X the number of intensifications.

From these tests with the vote systems, we can conclude that the SVM alone would be more beneficial than using a vote system. In both cases with ALS and ERP-S the SVM classifier was the best classifier at 10 intensifications. Figures 3.15 and 3.16 show the performances and progress of the votes and of SVM. During our experiment, we confirmed that with more intensifications the signal becomes better classified. The votes that were closer to SVM were V2 and V3 in the case of the ERP-S dataset with 84% and 83.8% respectively. With the ALS dataset the votes that were closer were V1 and V2 with 73.6% and 73.1% respectively. The V1 in the case of ALS had the same value as the SVM with 10 intensifications.

After we had chosen the SVM as the sole classifier we tried to improve its accuracy, by identifying the best Kernel for it. We tested two different kernels, the Radial Based Function (RBF) and the Normalyzed Poly Kernel (NPK). We decided to use these two kernels because they are the best kernels for SVM. RBF is a SVM algorithm that automatically determines centers, weights, and threshold that minimize an upper bound on the

expected test error. Poly Kernel represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models. These two kernels have parameters that need to be defined to optimize the results. We performed some tests to find their best values. For the RBF kernel we changed the value of its gamma and for the NPK the value of its exponent.

We did some tests with this two kernels to find the best combination to improve the recognition of P300. We tried multiples values for the NPK exponent: 5, 10, 15, 20, 25, 30, 40 and 50. For the RBF gamma we tried: 1, 5, 10, 20, 30, 40, 50. We chose these values to have a good range around the typical values used in the literature for these kernels. The results of our tests are represented in Tables 3.10 , 3.11, 3.12 and 3.13. We also present charts to compare the NPK and RBF kernels for each dataset, in Figures 3.17 and 3.18.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
Exp 5	66.4	**69.4**	67.5	71.1	71.3	72.8
Exp 10	66.6	69.3	**68.3**	**71.5**	71.6	74.3
Exp 15	66.0	69.1	68.1	71.3	**71.7**	**74.6**
Exp 20	**67.0**	68.3	68.1	71.1	71.5	74.3
Exp 25	66.5	68.6	68.1	71.4	71.3	74.3
Exp 30	66.5	68.4	67.6	71.1	71.3	73.4
Exp 40	66.1	68.2	66.0	70.8	71.0	73.4
Exp 50	65.3	67.7	65.0	70.4	70.9	72.1

Table 3.10: Accuracy results using NPK kernel with ALS dataset from 5 intensifications to 10 intensifications, for different exponent values.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
Gamma 1	**68.5**	**68.6**	**69.0**	**71.3**	**72.4**	**74.5**
Gamma 5	66.6	68.4	66.0	70.1	70.4	72.3
Gamma 10	64.4	65.4	64.2	67.0	69.3	71.3
Gamma 20	61.8	62.5	60.4	63.4	64.1	67.9
Gamma 30	59.3	59.9	59.6	58.6	60.4	60.3
Gamma 40	57.6	58.4	56.4	56.1	58.9	59.1
Gamma 50	58.1	55.9	54.1	55.4	57.4	55.8

Table 3.11: Accuracy results using RBF kernel with SVM using ALS dataset from 5 intensifications to 10 intensifications, for different Gamma values.

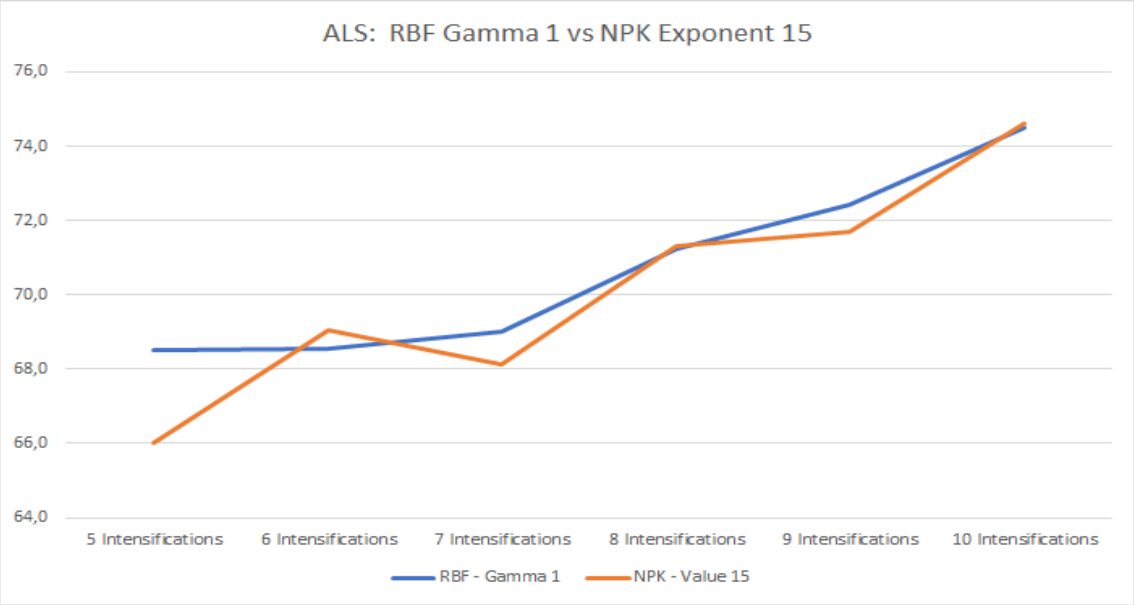


Figure 3.17: Chart of the results from the best combination that is RBF Gamma 1 and NPK exponent 15, from Table 3.10 and Table 3.11. Y is the accuracy and X the number of intensifications.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
Exp 5	74.7	**77.8**	77.8	**81.0**	78.6	84.3
Exp 10	75.3	77.2	78.4	80.6	**80.3**	**84.9**
Exp 15	**75.8**	76.8	**78.5**	80.6	**80.3**	84.8
Exp 20	75.6	76.8	**78.5**	80.9	79.9	84.6
Exp 25	75.3	76.3	78.2	80.8	78.7	84.0
Exp 30	74.8	76.2	77.4	80.5	79.3	83.5
Exp 40	74.9	75.6	76.5	80.2	79.3	84.3
Exp 50	73.9	74.4	76.6	80.2	79.8	84.2

Table 3.12: Accuracy results using NPK kernel with SVM using ERP-S dataset from 5 intensifications to 10 intensifications, for different exponent values.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
Gamma 1	**75.2**	**77.4**	**78.0**	**80.6**	**80.2**	**84.9**
Gamma 5	73.9	75.2	77.1	79.7	79.9	83.8
Gamma 10	73.3	72.9	76.1	77.2	77.5	80.3
Gamma 20	68.8	69.7	70.3	71.1	69.7	74.0
Gamma 30	63.6	65.6	60.9	66.9	65.0	68.4
Gamma 40	61.0	63.9	59.7	63.4	63.3	66.4
Gamma 50	61.2	59.9	57.6	61.1	62.5	60.2

Table 3.13: Accuracy results using RBF kernel with SVM using ERP-S dataset from 5 intensifications to 10 intensifications, for different Gamma values.

The best parameter for both datasets using NPK is using an exponent of 15 and for RBF is using gamma of 1. For the other cases we can see that the accuracy starts to

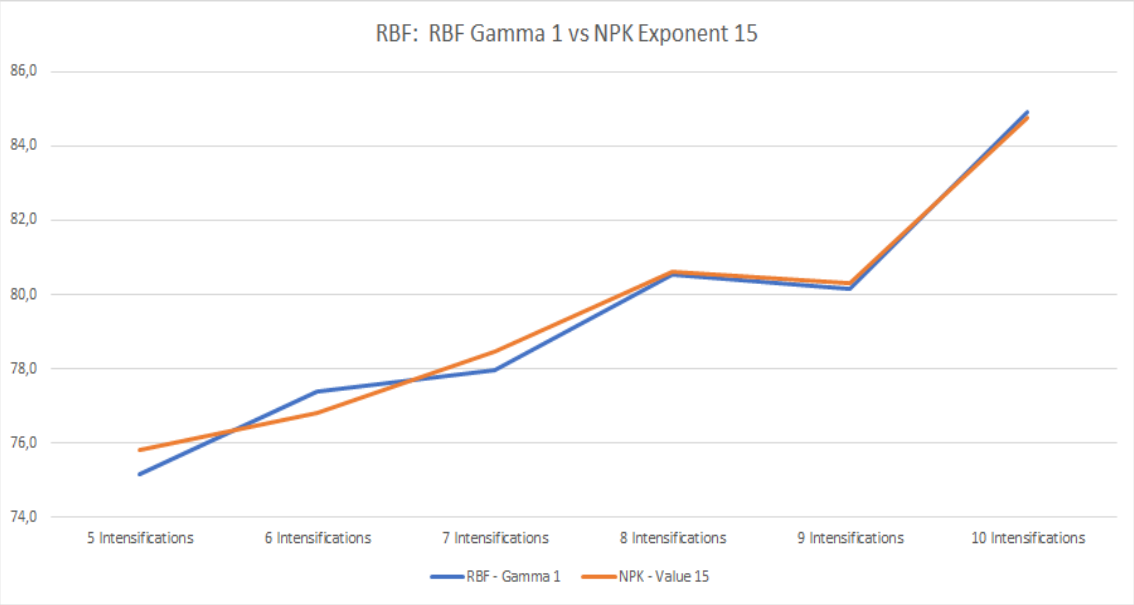


Figure 3.18: Chart of the results from the best combination that is RBF Gamma 1 and NPK exponent 15, from Table 3.12 and Table 3.13. Y is the accuracy and X the number of intensifications.

decrease. One specific case is with RBF, in which using gamma 1 and gamma 50 can jeopardize the classifier from 74.5% to 55.8%, for ALS, which means that the accuracy decreases almost 20%. In the case of NPK, it keeps almost the same value but decreases around 1% to 2%.

The results we had from the ALS datasets, presented in Table 3.10 and Table 3.11, lead us to believe that the best combination is the RBF using Gamma 1, because it presents better results along the intensifications and the best percentage of accuracy (74.5%) with 10 intensifications. Although, using the NPK with exponent of 15 and with 10 intensifications the accuracy is 75.6%, the results for the other intensifications makes us think it would not be the best because it presents lower values for 5 and 7 intensifications with 66% and 68.1% respectively. Meanwhile, the RBF with Gamma 1 presents 68.5% and 69% of accuracy for those number of intensifications. Figure 3.17 presents a chart where we can find the peaks of using NPK and the stability of using RBF.

With the ERP-S dataset the difference is not that much, it has very similar values using the same number of intensifications. The best result was with RBF with 84.9% of accuracy for 10 intensifications, while for NPK the accuracy was 84.8%. In Figure 3.18, we can see the progress of accuracy results with the increasing number of intensifications, and we can conclude that around 8 to 10 intensifications the results keep around the same.

In conclusion, we consider that the best classifier is SVM using RBF kernel with Gamma 1, because it presents a more stable classifier than using SVM NPK kernel.

3.3.6 Model Settings for P300 Detection

After the intensive set of tests, our resulting P300 Detection model consists in creating a signal by averaging all electrodes in a 1000 ms epoch size. After that we use multiples intensifications. From our results, we concluded that the best number of intensifications is 10, because it achieved the best accuracy of 74.5% for ALS and 84.9% for ERP-S. One key element in our model is normalization. By using normalization, our model can classify signals collected using different devices.

After feature selection, the best features to be used in our model are astchAB, astchAR, astchPR, bbchR, chR2, chR3, fillingR, eqR, a1, a2, a3, a4, r1, r2, r3, r4, MovY, quad2FillR, quad3FillR, quad4FillR, rectR1, rectR2, rectR3 and rms. The best classifier is Support Vector Machine with kernel Radial Basis Function and with property called Gamma set to 1.

3.4 Summary

In this chapter, we described our solution to detect P300 signals using geometric properties to describe the signal. First, we saw how we could distinguish P300 signals from Non-P300, which lead us to find differences between the two signals and find geometric properties to describe each one. The P300 signal has always a peak around 300 ms, or between 300 ms and 800 ms in the case of people with fatigue or others conditions, while the Non-P300 signal does not have. So, it is possible to detect P300 signals using its geometric properties.

We described a set of geometric properties used in a shape recognition called mCALI, which served as basis for our work. We added new features to the set to better describe the signals. At total we had 36 features at our disposal.

We described the EEG datasets used in our work, namely the ALS and ERP-S.

Our EEG signal pre-processing is constituted by an epoch of 1000 ms, to accomodate people who have their P300 long after the 300 ms. With our epoch we have more information to analyze. The second step is to create a single signal with the average of all electrodes used because having multiples signals from each electrode would be harder to classify. The third step is to create a signal by averaging intensifications. Each intensification used in this average had the 1000 ms epoch, the average of all electrodes used, and correspond to the intensification of a row or column of the speller matrix.

One important point for us was to be able to use signals from different devices. To that end, we explored the use of normalized signals. After the tests we chose to use normalized signals because we could create signals with the same amplitude, helping us to describe and classify them. So, for our signal pre-processing we added another step that is the normalization of the signal. With this new step we have four steps for our EEG signal pre-processing.

To discover the best features, we conducted a feature selection using Weka to help. The result from that feature selection lead us to use the 24 features presented in Table 3.4.

With these set of features for our model, our next step was to find the best classifier. We conducted tests using multiple classifiers such as SVM, Random Forest, AdaBoost, Bagging, BayesNet, LogitBoost and NaiveBayes, using their default parameters. The results from these tests revealed SVM had the best results, with accuracy of 73.6% at 10 intensifications with ALS dataset and 84.1% using ERP-S dataset. We also created some vote systems with the best classifiers but it did not improve the accuracy, so we stayed with SVM. To improve the accuracy we did some tests using SVM by changing its kernel and find the best parameters for it. We used two kernels: the Radius Basis Function (RBF) and Normalize Poly Kernel (NPK). The results from this test gave us the best classifier with 74.5% of accuracy for ALS dataset and 84.9% for ERP-S dataset, using RBF with the parameter of Gamma set to 1, which slightly improved the performance compared with SVM with default parameters.

With all of these steps we were able to identify the best features, classifiers and procedure that created our model for P300 Detection.

Chapter 4

P300 Detection using Central Moments

In this chapter, we describe our second approach for P300 detection using central moments. We describe the features used and the process behind each of them. We will also describe the EEG datasets, the pre-processing of the signal, the feature selection and the best classifier for it.

4.1 Central Moments

In this section we present the central moments, their purpose and their formulas. We explain the thought behind the formulas and how we intend to use them to classify the P300 signal.

In mathematics, central moments is a specific quantitative measure, used in both mechanics and statistics, of the shape of a set of points. The various moments form one set of values by which the properties of a probability distribution can be usefully characterized.

The formulas for Central Moments are:

$$Average = \frac{\sum_{i=1}^{n}}{n} \quad StandardDeviation = \sqrt{\frac{\sum_{i=1}^{n}(X-Average)^2}{n-1}}$$

$$Skewness = \sqrt{\frac{\sum_{i=1}^{n}(X-Average)^3}{n-1}} \quad Kurtosis = \sqrt{\frac{\sum_{i=1}^{n}(X-Average)^4}{n-1}}$$

The Average can be interpreted by the average of a collection of values. The Standard Deviation is a measure that is used to quantify the amount of the variance of the distribution. The Skewness measures how asymmetric the distribution is, and thus it gives information about the shape. The Kurtosis similarly to Skewness, provides information about the shape of the distribution. More specifically, Kurtosis is a measure of how flat or tall the distribution is in comparison to normal distribution.

From these four moments we will only use the Average and Standard Deviation. These two moments are the best to describe the signals because it can shows us the distribution of the values from the signal, helping us to classify.

4.2 P300 Central Moments

The reason for using the central moments is that they can be used to compare collections of values, so we intend to use that in our favor. In the signal, there are parts where the P300 and other components of the signal appear, as mention in Chapter 2. So, we divided regions of the signal based on these components so that we could describe the signal and classify it. Some of the components are N2, P2, P300 and other components. In some regions the values of a P300 signal and Non-P300 signal can differ. In the case of P300, it has the highest peak around 300 ms and 400 ms, while in a Non-P300 signal it does not have a highest peak. So, a P300 signal will have higher values than a Non-P300 and its average and standard deviation will be different from a Non-P300 signal.

Table 4.1 represents the regions of the signal that we consider to be the best to describe the P300 and Non-P300 Signals.

Region Name	Start	End	Reason
Rg1	150	250	The P2 and N2 that are the second highest and lowest peaks in the signal after the stimulus, are in this interval.
Rg2	200	300	The N2, which is the second lowest peak before the P300, is in this interval.
Rg3	300	400	The P300 happens here.
Rg4	400	500	In case the P300 happens later than the normal (fatigue or not focused enough).
Rg5	500	600	
Rg6	600	700	
Rg7	250	600	From the N2 to get the P300.
Rg8	400	600	After the P300 in case the P300 happens later.

Table 4.1: All regions of the signal with importance.

We tried to include all tiny regions of the signal, where it might create differences and describe a P300 and Non-P300 signal. In Figure 4.1, we present the regions chosen to our model, which were described in Table 4.1. Rg1, between 150 ms and 250 ms, includes the P2 that is the second highest peak and N2 with the second lowest peak before the P300. Rg2, between 200 ms and 300 ms, happen the N2 and the beginning of the P300. Rg3, between 300 ms and 400 ms, is where normally happens the P300. The Rg4, Rg5 and Rg6, between 400 ms to 500 ms, 500 ms to 600 ms and 600 ms to 700 ms, respectively, are some regions in which the P300 can happen later than the normal time, because some people have a lower reaction, fatigue or others problems that affect the P300. Hence, every person is different and every signal is different from person to person. We also chose two bigger areas, Rg7 and Rg8, between 250 ms to 600 ms and 400 ms to 600 ms, where we try to take almost all the signal to see the difference between tiny regions and larger regions.

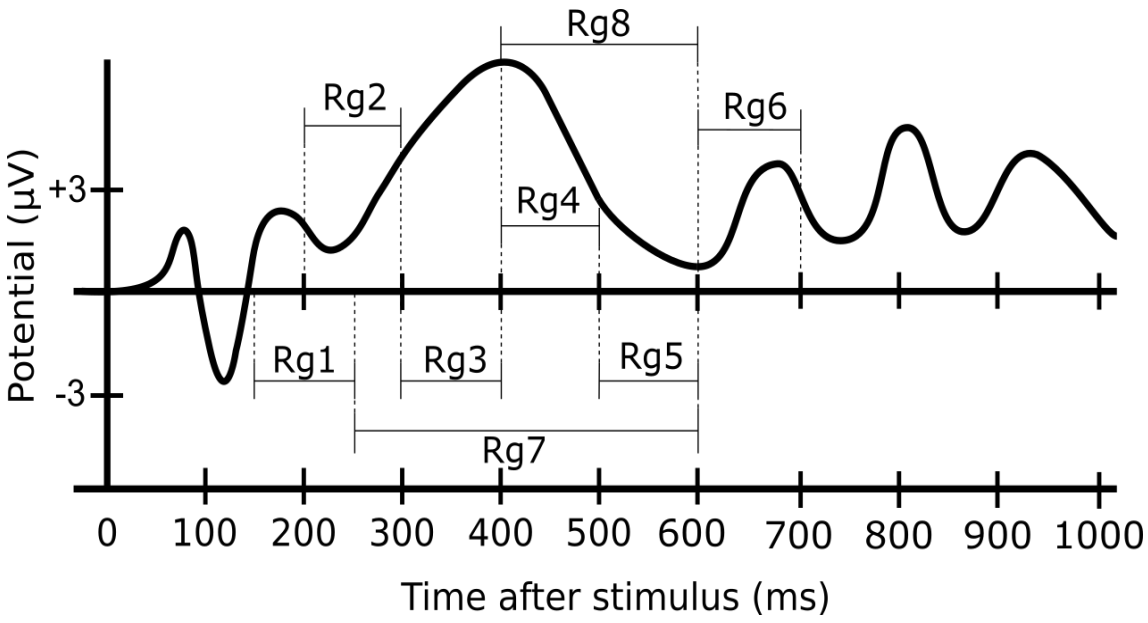


Figure 4.1: P300 signal with the regions presented in Table 4.1.

4.3 Model Settings using Central Moments

In this section, we describe the EEG datasets, the EEG pre-processing, the feature selection and the classifier used for testing and the final model setting.

4.3.1 EEG datasets, EEG Pre-Processing and Feature Selection

The EEG datasets used here were the same used in the previous method of P300 Detection, which are the ALS and ERP-S datasets.

The EEG pre-processing is the same as used in the Geometric Detection: an epoch of 1000 ms, the average of the electrodes used, the average of intensifications and finally the normalization of the signal.

In the case of the feature selection, we had 16 values to work with. Using the same approach as in the geometric detection model, we used Weka to do a feature selection, with the same methods. We used GreedStepwise as search method and CfsSubsetEval as attribute evaluator and using Ranker with the attribute evaluator CorrelationAttributeEval to validate the results. From the obtained results, there was no need for feature selection, since almost all results used all the 16 values created for the 8 regions we had defined. So, our model will use the 16 features mentioned.

4.3.2 Classifier

To choose the best classifier, we tested using the same method of Geometric Detection, that is leaving one of the users and train with the rest and evaluate with the user left behind, changing the order so that every user is used in the evaluation.

We did not do an intensive search for the best classifier like we did with the other detection method. Therefore, we decided to evaluate our detector with the final classifier from our intensive search for the Geometric Detection (SVM RBF with Gamma 1) and the other classifier with the second best accuracy (Random Forest). The results for both classifiers are displayed in Tables 4.2 and 4.3, using the same datasets ALS and ERP-S.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
SVM RBF	**69.5**	68.3	**71.9**	**72.6**	**74.1**	73.1
Random Forest	69.1	**68.5**	71.1	**72.6**	72.6	**73.5**

Table 4.2: Accuracy results from two classifiers using ALS dataset from 5 intensifications to 10 intensifications.

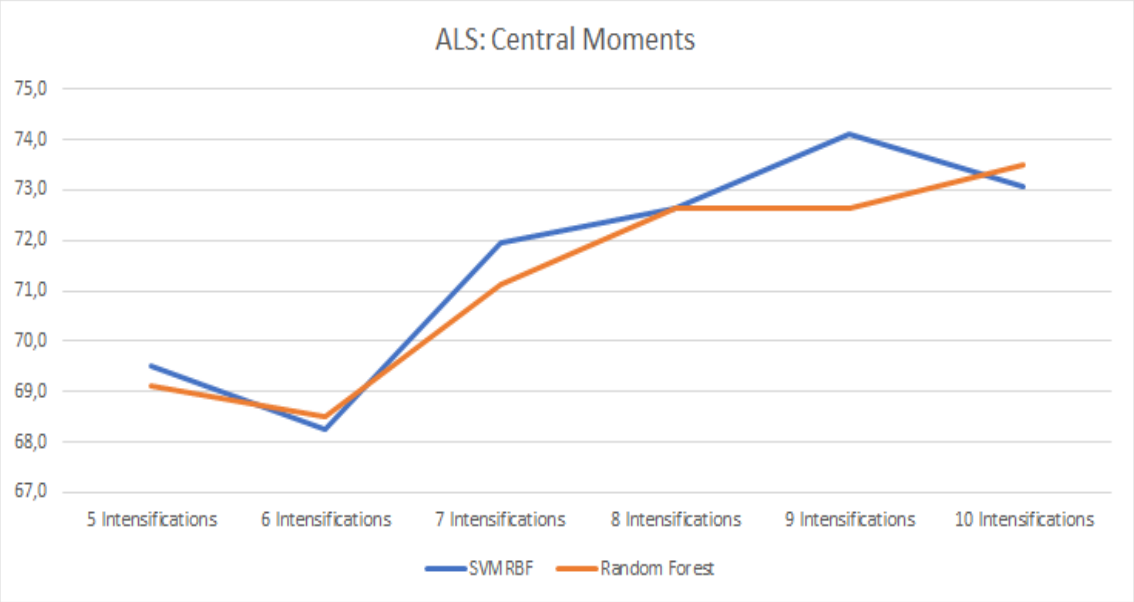


Figure 4.2: Chart of the results using the best classifiers, SVM and Random Forest with ALS data. Y is the accuracy and X the number of intensifications.

	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
SVM RBF	**82.1**	**81.8**	83.3	**86.4**	85.7	**86.0**
Random Forest	81.2	81.4	**83.7**	85.6	**85.8**	85.4

Table 4.3: Accuracy results from two classifiers using ERP-S dataset from 5 intensifications to 10 intensifications.

Both classifiers presented around the same accuracy in both datasets. With 10 intensifications for ALS dataset, the best classifier was Random Forest with 73.5% of accuracy. Meanwhile, SVM RBF had an accuracy of 73.1%, with a difference of 0.4%. However, with ERP-S dataset the best classifier was SVM RBF with 86% of accuracy, hence with Random Forest the accuracy was 85.4%, with a difference of 0.6%. Figures 4.2 and 4.3

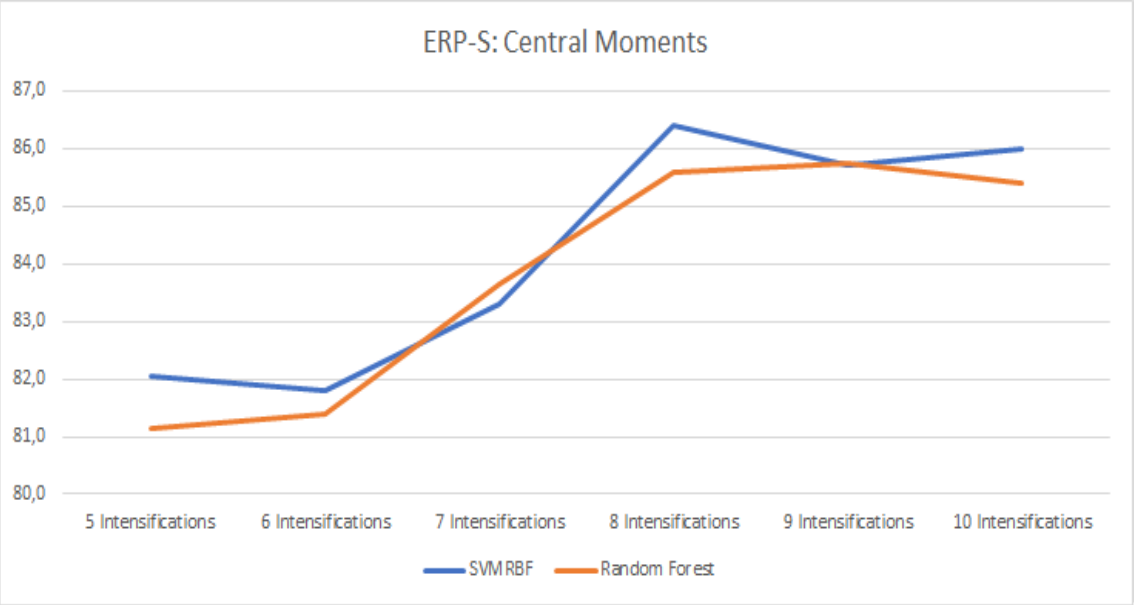


Figure 4.3: Chart of the results using the best classifiers, SVM and Random Forest with ERP-S data. Y is the accuracy and X the number of intensifications.

present the behaviour of accuracy performance when increasing the number of intensifications. From the obtained results we chose the same classifier used with the Geometric Detector that is SVM RBF, because it presented the best results and stability.

4.3.3 Model Settings for P300 Detection using Central Moments

Our model creation of P300 Detection lead us to the same steps of the Geometric approach, with the first step being the creation of the signal by averaging all electrodes, with 1000 ms of epoch. After that we use multiples intensifications and create a normalized signal to train and classify. The chosen classifier was the same: SVM RBF with gamma 1. It is the best classifier, because it is more stable, has 73.5% of accuracy for ALS and 86% for ERP-S. The features used are based on the average of the signal and the standard deviation of the same signal, computed by regions. The chosen regions to describe the P300 signal and where we can find the fundamental differences to a Non-P300 signal. We created 8 regions distributed between 150 ms and 700 ms of the signal. These regions are positioned in sections where ERP components happen, like N2, P2 and P300. Some of these components are pre-P300 and so we can create the best description of the signal.

4.4 Summary

We first defined what are central moments, explaining that they are measures of the shape of a set of points. From the central moments we chose two moments that best describe the signals: Average and Standard Deviation.

From those formulas we explained their use in the signal, because in some regions of the signal we can find components like N2, P2 and P300, allowing us characterize P300 and Non-P300 signals. From those regions we created 8 regions to describe the P300 and Non-P300 Signals, which are positioned between 150 ms and 700 ms.

We used the same datasets (ALS and ERP-S) using the same method of testing, leaving one user out to evaluate it. Also, we used the same pre-processing with four steps. The first step is the creation of an epoch with size of 1000 ms. The second step is the creation of the signal with all electrodes used after the stimulus. The third step is a new creation of a signal with a number of intensifications. Lastly, we normalize the final signal.

We conducted a feature selection that lead us to keep all the initial features from the 8 regions, resulting in 16 features.

We then conducted tests to determine the best classifier using Random Forest and SVM RBF. The results from these tests lead us to use the SVM RBF because it presented the best results with accuracy of 73.1% for ALS dataset and 86% for ERP-S dataset.

Chapter 5

Experimental Evaluation

In this section, we evaluate our models for P300 detection using geometric features and central moments. We describe the evaluation method, and present and analyze the results. We compared our models between them using several datasets, and we also compared them with the approach used in [Riccio et al., 2013] and with Peak Picking.

5.1 EEG Datasets and Experimental Procedure

In this section we describe the experimental procedure, presenting the models under evaluation and the tests performed.

5.1.1 EEG datasets

The datasets used were the same used to generate our models: ALS and ERP-S. Additionally, we used another dataset from the same authors of the ERP-S, the GEO dataset (see Section 3.3.1).

5.1.2 Models for Evaluation

For the evaluation we used our two models created to detect P300 signals, and compared them with two other models for detecting P300. One was the normal P300 detector Peak Picking, that we implemented. The other model was from [Riccio et al., 2013], which we have the results of their models.

For the implementation of the Peak Picking we created a threshold between all P300 signals that were considered target used to train the model. From the signals we started by getting the highest positive value (P300) between 250ms and 600 ms and the negative value (N1) between the 0ms and the 250ms of the signal. From this two values we subtracted the P300 with N1 and the result was put in an array and ordered with the results of the other signals used to train. With all values ordered we selected a value from the array to be our threshold. The selected value was picked from the box plot of the array and

selected the first quadrant to be the threshold. To evaluate the method we conducted the same process to the signals to be evaluated and the result was compared with our threshold. If the value was bigger than the threshold, then we considered it as P300, otherwise it was a Non-P300.

5.1.3 Tests

We conducted 3 different evaluations. The first evaluation was user-independent using our two models and Peak Picking. The process for this test was the same used to create our models, by using the leave one out method, that is, training the model with all users minus one, and then use this one for validation as explained before (Section 3.3). For this test we used only ALS and ERP-S datasets, because we wanted to use two different kinds of signals amplitude to test our models.

The second evaluation was user-independent but with crossing datasets: training the models with one dataset (Example ALS dataset) and evaluating with another dataset (Example ERP-S dataset). With this test we intended to see how our models behaved with different signals used to train and to evaluate. The models used in this test were our two models (Geometric and Central Moments models) and we used all the datasets available (ALS, ERP-S and GEO datasets).

In the third evaluation we compared our models with Peak Picking and Riccio's model. To compare our models with Riccio's model we conducted the same process they used, which was user-dependent with seven-fold cross-validation, meaning that they used signals from the user to train their model and evaluate with signals from the same user. Thus, we conducted the tests using the ALS dataset. Additionally, we conducted the same process with the other datasets to compare our model with the Peak Picking.

5.2 Results of Evaluation

In this section, we present the results from the experimental evaluation.

5.2.1 User Independent

Tables 5.1 and 5.2 present the results of our models in comparison with the Peak Picking P300 detector. For each classifier we show the results achieved for the dataset, by number of intensifications.

Classifier	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
Geometric	68.5	**68.6**	69.0	71.3	72.4	**74.5**
CM	**69.5**	68.3	**71.9**	**72.6**	**74.1**	73.1
PP	54.3	54.3	54.3	54.4	57.4	56.3

Table 5.1: Accuracy results for all models using ALS dataset, from 5 intensifications to 10 intensifications.

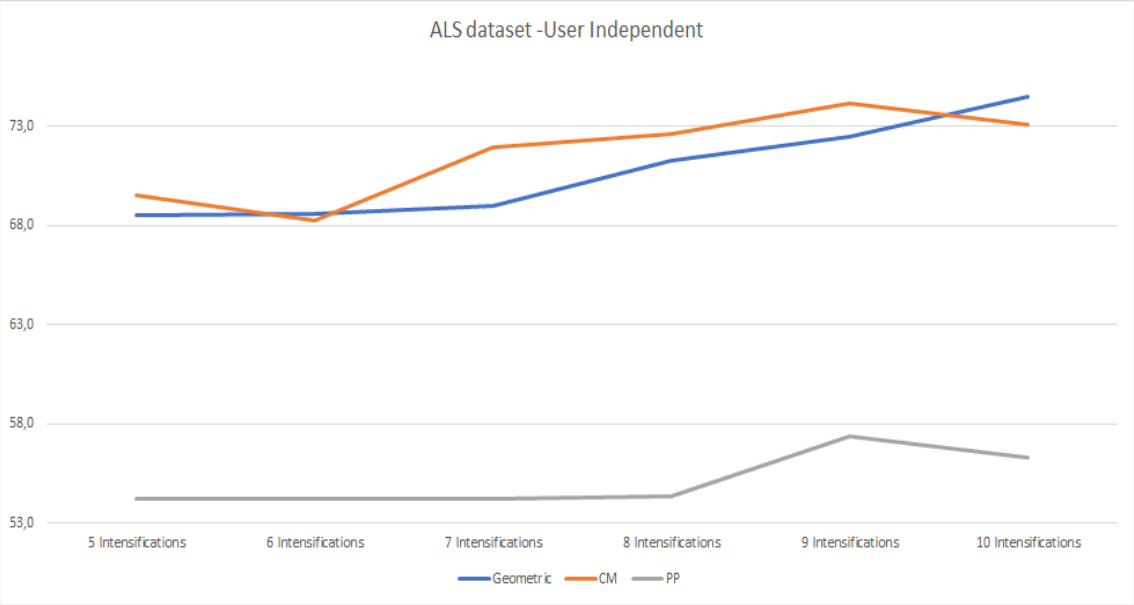


Figure 5.1: Chart of the Table 5.1. Y is the accuracy and X the number of intensifications.

Classifier	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
Geometric	75.2	77.4	78.0	80.6	80.2	84.9
CM	**82.1**	**81.8**	**83.3**	**86.4**	**85.7**	**86.0**
PP	73.2	72.7	75.2	75.1	75.7	76.9

Table 5.2: Accuracy results for all models using ERP-S dataset, from 5 intensifications to 10 intensifications.

The results show that the central moments model is better and more stable than the others, while using the ERP-S dataset. In Figure 5.2 we can see the difference between the models with the central moments model having always better results, achieving an accuracy of 86% with 10 intensifications. However, with the ALS dataset the geometric model has almost the same results as the central moments model. Hence, it has the best result for 10 intensifications with 74.5%, while the central moments has an accuracy of 73.1%.

The peak picking in the case of ALS is almost like a coin flit. However, with ERP-S the results are better with 76.9% for 10 intensifications. This results show that the ALS

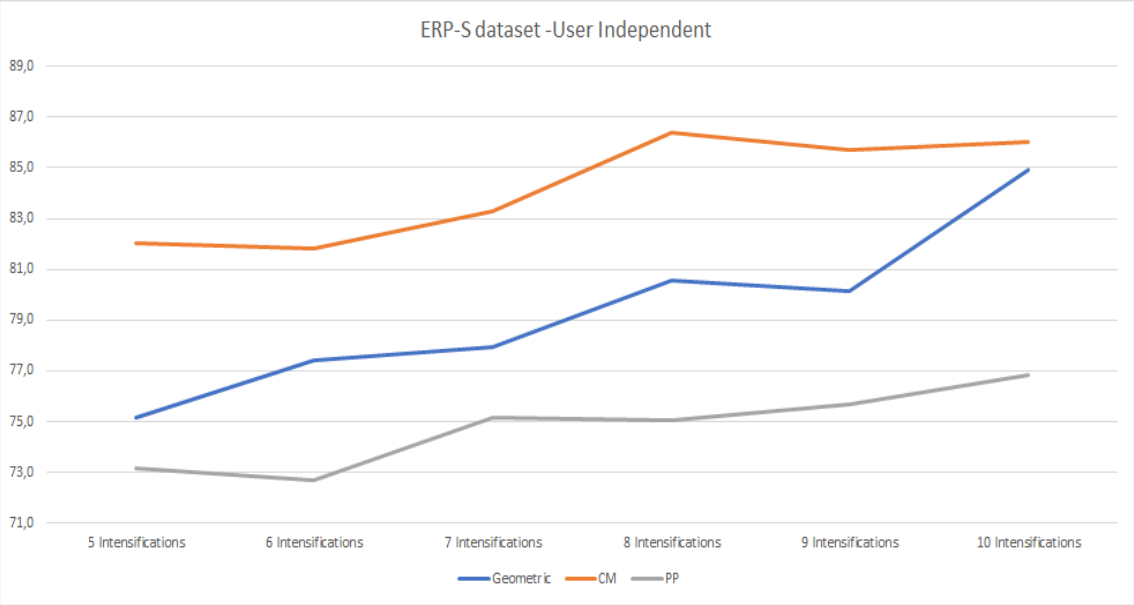


Figure 5.2: Chart of the Table 5.2, Y is the accuracy and X is the number of intensifications.

dataset is a hard dataset for detection, because it presents values around the same size for P300 and Non-P300, which leads to classification problems. Despite that, our models were able to classify the signals and achieve good results as presented in Figure 5.1.

5.2.2 Dataset vs Dataset

Tables 5.3, 5.4 and 5.5, present the results for training with one dataset and evaluating with another, using ALS, ERP-S and GEO datasets respectively. The tables are organized by training dataset, evaluation dataset, classifier used and accuracy by number of intensifications.

Train	Evaluate	Classifier	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
ALS	ERP-S	Geom	**67.5**	**67.8**	65.6	**71.7**	**70.5**	**75.3**
ALS	ERP-S	CM	61.8	63.9	**68.6**	69.4	70.0	69.2
ALS	GEO	Geom	67.8	68.2	69.9	72.6	72.3	72.1
ALS	GEO	CM	**72.1**	**72.3**	**75.6**	**75.9**	**74.7**	**76.1**

Table 5.3: Accuracy results when using ALS as training set and ERP-S and GEO to evaluate, using Geometric and Central Moments models.

When training with ALS dataset, the best results were achieved with the GEO dataset (Figure 5.3) using both models.

When using GEO as the evaluation dataset, the best model overall was CM, presenting always better accuracies than the Geometric model, achieving 76.1% for 10 intensifications. However, when using the ERP-S dataset the best model is the Geometric, with an accuracy of 75.3% for 10 intensifications, while the CM has an accuracy of 69.2%.

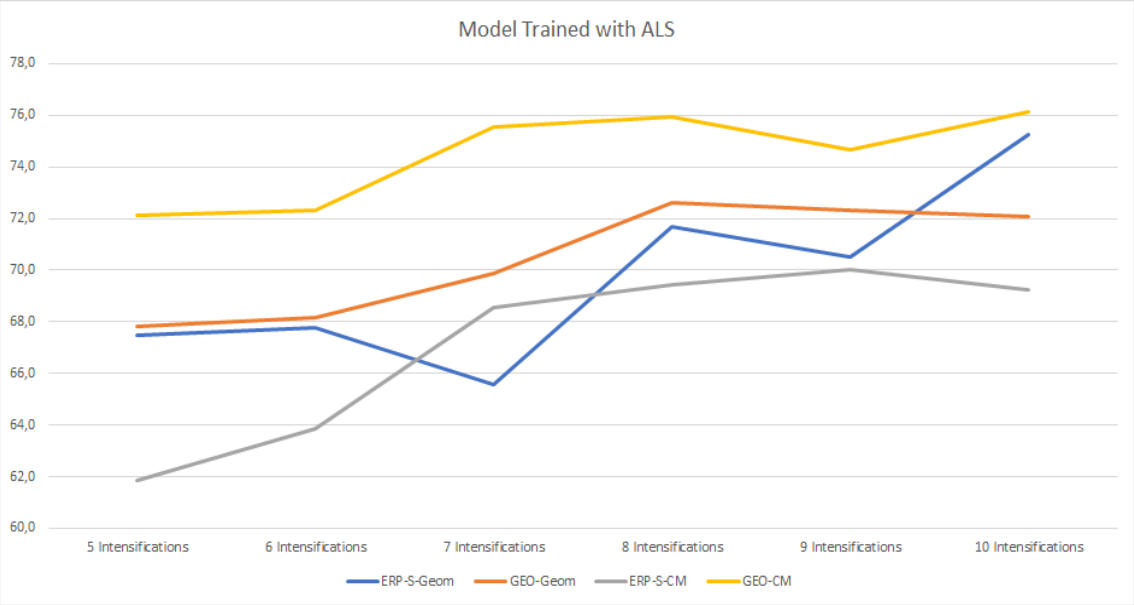


Figure 5.3: Chart of the results from Table 5.3, Y is the accuracy and X is the number of intensifications.

From this results we can conclude that the ALS dataset allow the creation of models that give good results with any kind of dataset. Comparing with the results we had when training with ALS and evaluation with ALS as well, the results are very close.

Train	Evaluate	Classifier	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
ERP-S	ALS	Geom	**55.7**	**57.6**	**56.4**	**59.6**	**56.1**	57.8
ERP-S	ALS	CM	55.5	56.1	55.4	59.3	55.7	**59.3**
ERP-S	GEO	Geom	72.2	75.8	77.5	78.7	79.0	77.8
ERP-S	GEO	CM	**77.9**	**76.6**	**78.7**	**80.4**	**82.7**	**85.0**

Table 5.4: Accuracy results when using ERP-S as training set and ALS and GEO as evaluate using Geometric and Central Moments models.

The results using ERP-S as training model, reveals that it works better when it is evaluated with signals of the same kind, like the GEO dataset that was recorded from the same device. It achieved results higher than 70%, with the highest accuracy of 85% with 10 intensifications using the central moments model. Hence, when evaluating with the ALS dataset the results are worst, with values below 60%. From Figure 5.4, we can see that the central moments model presents the better results when evaluating with the GEO dataset, outperforming the geometric model. Meanwhile, when using the ALS dataset the Geometric model presents the better results. However, for 10 intensifications the central moments achieves 59.3%, while the geometric model has 57.8%.

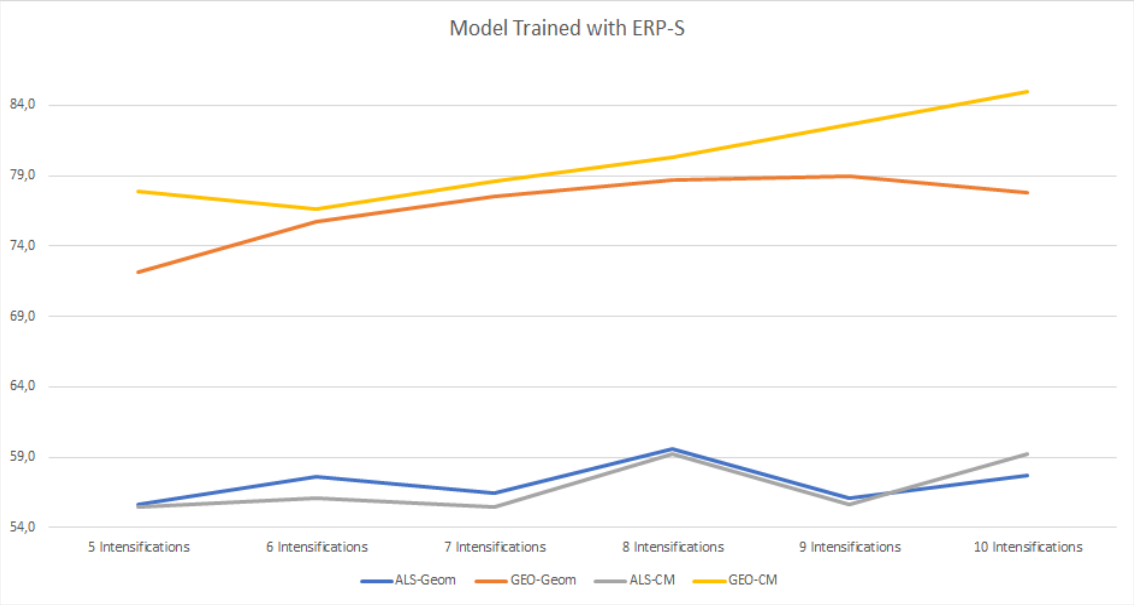


Figure 5.4: Chart of the results of Table 5.4, Y is the accuracy and X is the number of intensifications.

Train	Evaluate	Classifier	5 Int	6 Int	7 Int	8 Int	9 Int	10 Int
GEO	ERP-S	Geom	75.3	78.9	80.3	81.8	82.3	**84.5**
GEO	ERP-S	CM	**78.4**	**79.0**	**81.7**	**82.2**	**82.8**	**84.5**
GEO	ALS	Geom	57.3	55.9	56.2	58.8	56.6	58.2
GEO	ALS	CM	**63.1**	**61.5**	**62.3**	**65.4**	**63.4**	**64.7**

Table 5.5: Accuracy results when using GEO as training set and ERP-S and ALS to evaluate, using Geometric and Central Moments models.

From the results using the GEO dataset as training model (Table 5.5), when using the ERP-S to evaluate it presents the better results with the CM model for all intensifications. However, with 10 intensifications it presents the same accuracy as the Geometric. When using the ALS for evaluation the results are close to what happened when we trained with ERP-S and evaluated with ALS (Figure 5.5). However, when using the CM model the results are a bit better, being between 61% and 65%, which is higher than using the geometric classifier.

In conclusion, based on Figures 5.4 and 5.5 we can see that when using ALS as evaluation dataset the results are poorer which means that its signals are hard to detect. However, in Figure 5.3, when using ALS as training model the results using both ERP-S and GEO for evalution are very similar and better.

Overall, we can conclude that for user-independent P300 detection, the CM model is better than the Geometric model.

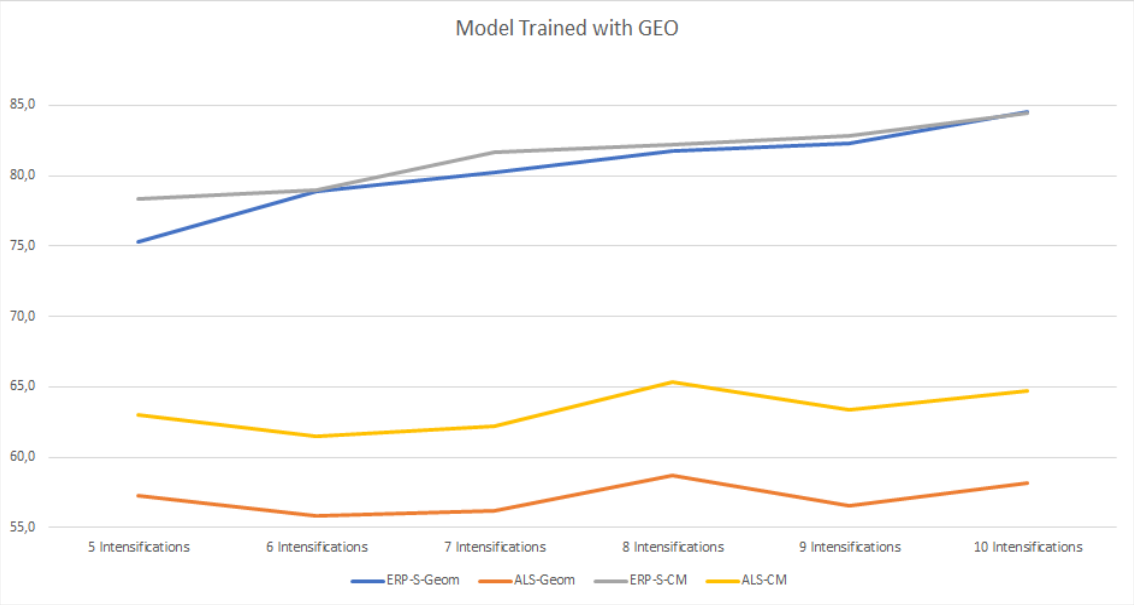


Figure 5.5: Chart of the results of Table 5.5, Y is the accuracy and X is the number of intensifications.

5.2.3 User-Dependent

Table 5.6, shows the results of our seven-fold cross-validation with ALS dataset using the Geometric model, Central Moments model, Peak Picking model and finally the results of the authors of the dataset ALS, which used SWLDA as classifier [Riccio et al., 2013].

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8
Geometric	62.9	65.7	77.1	72.9	83.6	81.4	69.3	80.7
Central M.	67.1	**88.6**	67.1	75.7	**87.1**	86.4	87.9	88.6
Peak Picking	50.0	52.1	49.3	55.7	55.0	62.1	68.6	67.9
SWLDA	**84.5**	86.3	**87.2**	**85.9**	86.2	**88.6**	**88.6**	**92.3**

Table 5.6: Accuracy results from user dependent test using ALS dataset.

From the results obtained using the ALS dataset, we can conclude that using our models does not reach the same accuracy of the SWLDA that is always above 84.5%. However, the only model with better accuracy in two subjects was the Central Moments model, with 88.6% and 87.1% for user 2 and user 5, respectively, while with SWLDA they had 86.3% and 86.2% of accuracy respectively (Figure 5.6). In comparison with our models the Peak Picking presents terrible results, in one case below 50%. The geometric model presented some good results that were close to the CM results. However, it did not present great results only satisfactory results.

Tables 5.7 and 5.8, shows results of seven-fold cross-validation we conducted for ERP-S and GEO dataset, however with only three models (Central Moments, Geometric

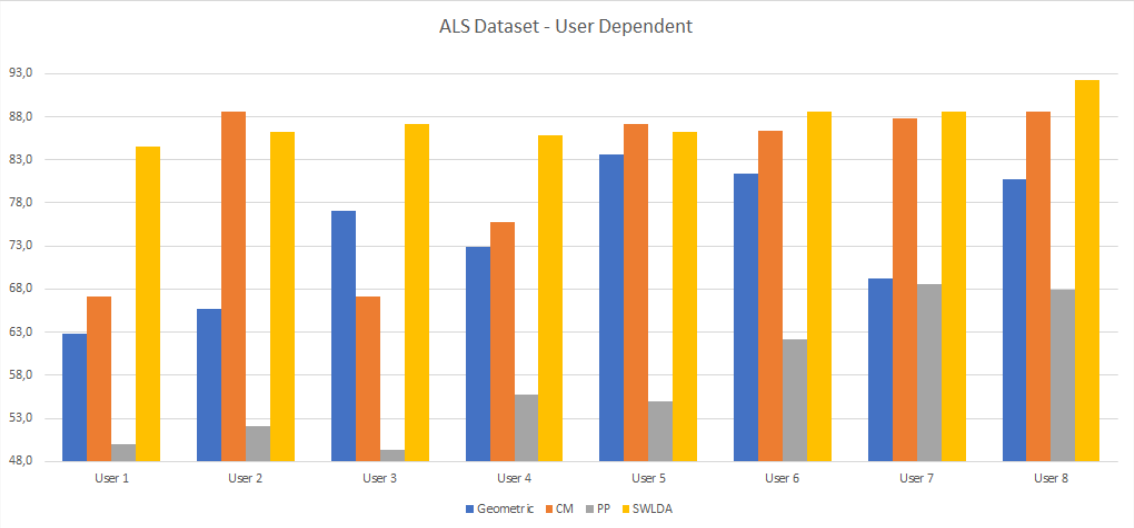


Figure 5.6: Chart of bars for user dependent test from Table 5.6, Y is the accuracy and X is the user.

and Peak Picking), because the Riccio's model was only tested with the ALS dataset, and we did not have their model for testing.

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10
Geometric	87.9	87.9	80.6	75.9	75.9	**80.7**	73.3	71.1	**97.6**	87.9
Central M.	**92.7**	**92.7**	**92.7**	**78.3**	**85.4**	66.2	**78.3**	**87.9**	**97.6**	**92.8**
Peak Picking	73.6	76.0	61.7	61.6	76.0	64.1	64.1	75.9	75.9	73.6

Table 5.7: Accuracy results from user dependent test using ERP-S dataset.

The results from the ERP-S dataset (Table 5.7) the CM model is better than the Geometric model. The CM model presented the best accuracy with user 9, obtaining 97.6%, and with user 1, 2 and 3 the accuracy was 92.7% (Figure 5.7). However, only in one user the Geometric model presented better results with 80.7% of accuracy for user 6, while using the CM model the accuracy was 66.2%, which is the lowest accuracy from the CM model. The Peak Picking had a stable accuracy between 61% and 76%.

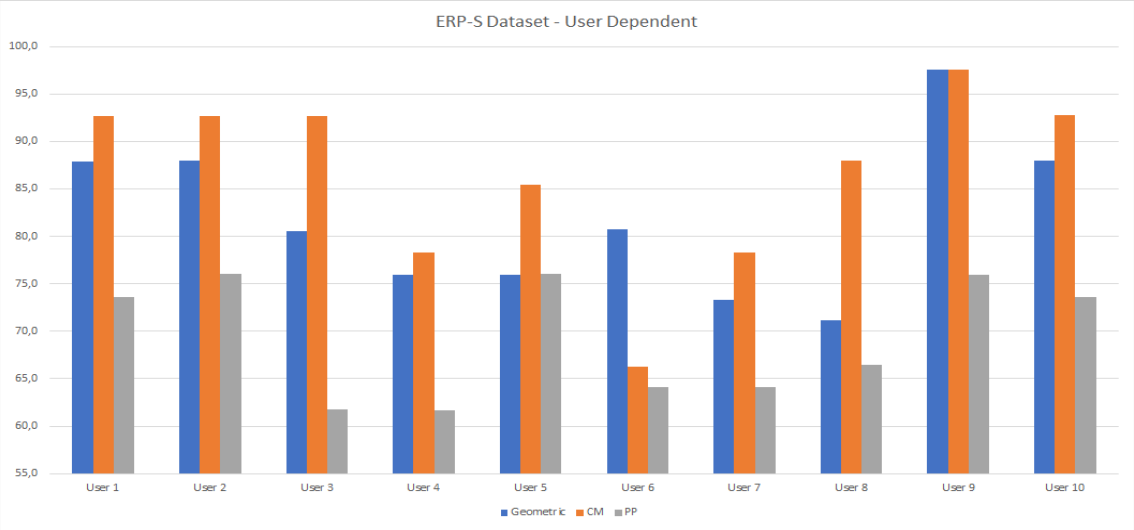


Figure 5.7: Chart of bars for user dependent tests from Table 5.7, Y is the accuracy and X is the user.

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10
Geometric	78.4	83.1	78.2	73.5	68.7	73.4	78.2	75.9	83.1	85.4
Central M.	**90.4**	**100**	**95.1**	**80.6**	**78.2**	**73.5**	**83.0**	**87.9**	**92.7**	**92.7**
Peak Picking	68.8	73.6	78.4	71.2	64.1	68.8	61.6	75.9	73.6	71.2

Table 5.8: Accuracy results from user dependent test using GEO dataset.

The last test using GEO dataset (Table 5.8), the conclusion is the same as the last two. CM model presented the best accuracy in all users tested, having in one case 100% of accuracy with user 2. Meanwhile, geometric model rounded between the 73% and 85%, not achieving results as good as the CM model (Figure 5.8). The Peak Picking presented the same stable accuracy as in the last test with the ERP-S dataset, which is between the 61% and 78%.

From these results we can conclude that our models may not reached the same results as the SWLDA for the ALS dataset, but, when using other datasets we achieved good results with accuracies higher than 90%. One conclusion that we can take from these tests is that the CM model is better for user-dependent situations than the geometric model. This means that using the CM model as a learning classifier from the user may reach better results. However, the geometric model sticked with the same results from the past tests demonstrating a stable accuracy that rounds the same values above 70% and lower than 80%. The Peak Picking presented bad results for the ALS dataset but a more stable accuracy for EPR-S and GEO dataset. This proves what we verified before that the ALS

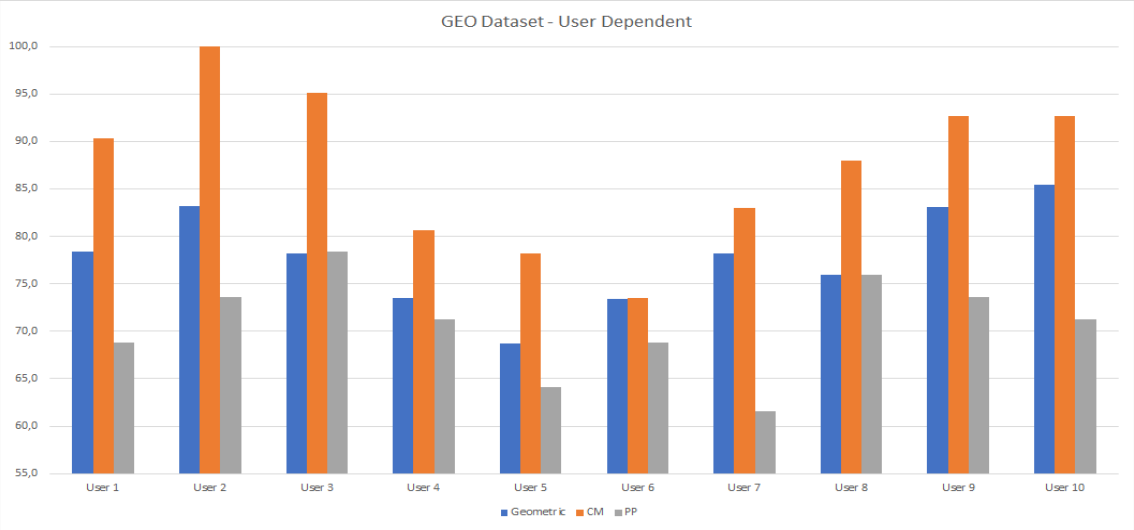


Figure 5.8: Chart of bars for user dependent tests from Table 5.8, Y is the accuracy and X is the user.

dataset presents signals that are hard to classify, while ERP-S and GEO presents better signals.

5.3 Discussion

From our tests we could understand the behaviour of our models, Geometric Model and Central Moments Model, in different environments: i) user-independent, where we used the leave one out method; ii) user-independent, but with crossing datasets, where we trained the models with one dataset and evaluated with a different dataset; iii) user-dependent, with seven-fold cross-validation to train and evaluate with signals from the same user.

In terms of user independent we compared our two models, with a version of the Pick Picking P300 detector implemented by us. From those tests we verified that the Central Moments Model presented better results and stability than the Geometric Model when using better signals (ERP-S). However, when using bad signals (ALS) both of them showed close results, demonstrating that both of them when trained with bad signals still can classify. For Peak Picking the results from using bad signals (ALS) were almost like a coin flip, hence when using good signals (ERP-S) the algorithm presents a reasonable accuracy.

When training with one dataset and use an other dataset to evalue, we found that our models can still classify when using bad signals (ALS) as training model. However, when using good signals (ERP-S and GEO) as training model and using bad signals to evaluate the results are worst. Hence, when using good signals when evaluating the accuracy is far better than using bad signal to train the model. The model that presented better results

overall was the Central Moments Model.

In terms of using signals from the user as training model and to evaluate, we first compared our models and the Peak Picking with the results from the authors of the ALS dataset, which used SWLDA as classifier. The results showed that SWLDA had better accuracy overall compared with the others models. However, the Central moments presented a close accuracy to the SWLDA, even having better results for two users. The geometric model achieved a stable accuracy but not good enough as the CM model. The Peak Picking had a 50% accuracy, which proves our point of ALS dataset being a bad dataset. When using the ERP-S and GEO datasets, the results were far better compared to the ALS dataset. The Central Moments model presented again the best results in both tests. The results achieved on these test were far better than the user-independent and Dataset vs Dataset, which means that the CM model is good when we are using signals from the same user.

From these tests we can conclude that the CM model presents better results than the Geometric. However, in some cases Geometric was close to achieve the same results but it was not enough.

5.4 Summary

In this chapter, we analyzed the performance of our models in three test environments: i) User independent that is using signals from the same dataset, leaving one of the users to be used as evaluation signals and using the rest for training the model, changing the order of the users; ii) use of different datasets as training model and use others for evaluation; iii) user dependent, where we did a seven-fold cross-validation using signals from the user and evaluating with other signals from the same user. We used 4 models to evaluate, the CM model, Geometric model, Peak Picking and the last model (only used for User-dependent) from the authors of the ALS dataset, SWLDA.

From the first test, the results showed that CM model was a more stable classifier than the Geometric Model and it had better results in both datasets. Meanwhile, using Peak Picking the results were very poor using the ALS dataset, but using ERP-S the results were better but not great.

From the second test, we concluded that the ALS dataset used as training model is good because using signals from the same dataset or using different signals the results are good. However, when used as evaluator the results are bad when the model is trained with different signals. From this tests we conclude that CM presented the better results. Only in one test the Geometric model had a better result.

For the last test, we conducted the user-dependent test with seven-fold cross-validation for the ALS dataset. The results we had from our models were not enough to have a better and stable classifier than the SWLDA, only the CM had better results with two users. For

the other datasets, we had better results overall, with CM being the best classifier with most of his results higher than 90% and in one case 100%.

Overall, the Geometric model looks to be a stable model that adapts with every environments keeping the same accuracy around 70% and 80%. The CM model still presents the better results to the same tests but outperforms the Geometric model in User-Dependent, with results above 90%.

Chapter 6

Conclusions and Future Work

In this chapter, we present the main conclusion of the developed work, the future work to be done, and some limitations and ideas to improve our models.

6.1 Summary of Dissertation

In this work we presented two models for detecting the P300 signals, the Geometric Model and the Central Moments Model. The models created support user-independent and user-dependent scenarios. We compared both models to find which one is better.

In chapter 2, we described what is a P300 signal, how to generate it and its usefulness. We then described approaches to detect P300 signals. We also described Spellers that are used to visually the signals to classify.

In chapter 3, we presented our Geometric model, that uses geometric properties to describe and identify P300 Signals. We presented how we could detect P300 signals with its geometric properties and presented a set of features that best describe the signals (P300 and non-P300). We then created our model by conducting a set of tests to find if it was worthy to use normalized signals and the best classifier for our set of features.

In chapter 4, we presented our second model that uses Central Moments (Average and Standard Deviation) in certain regions of the signals to describe and identify P300 and Non-P300 Signals. We conducted the same tests to find the best classifier for that collection of features.

In chapter 5, we conducted our experimental evaluation where we compared our two models to find which one was better, by conducting 3 evaluations, one using user-independent, using different datasets and lastly conducting a user-dependent tests. From these tests we found that the best model was the Central Moments model. In the first two tests, they presented similar accuracies, however Central moments presented better results. In the last test the Central Moment outperformed the Geometric Model with an accuracy higher than 90%, and in one case 100%.

6.2 Contributions and Limitations

In the end of this dissertation, we have two new models to detect P300 using geometric properties and central moments. With these models we contributed for the small sample of classifiers that uses the shape of the signals to describe and identify P300 in EEG Signals.

The limitations of our models is that they do not present the same accuracy as the best classifiers to detect P300 and still need some training to achieve good results.

6.3 Future Work

For future work, we believe that using this kind of approach based on the shape of EEG signals may lead to other models.

In the case of our models, to improve them we could join the two models into one and conduct the same test we did to see if it improves or not. Other possible approach is adding new geometric features to the geometric model that may increase its accuracy and overcome the Central Moments model.

Bibliography

[Akram et al., 2015] Akram, F., Han, S. M., and Kim, T.-S. (2015). An efficient word typing p300-bci system using a modified t9 interface and random forest classifier. *Computers in biology and medicine*, 56:30–36.

[Aloise et al., 2012] Aloise, F., Aricò, P., Schettini, F., Riccio, A., Salinari, S., Mattia, D., Babiloni, F., and Cincotti, F. (2012). A covert attention p300-based brain–computer interface: Geospell. *Ergonomics*, 55(5):538–551.

[Alvarado-González et al., 2016] Alvarado-González, M., Garduño, E., Bribiesca, E., Yáñez-Suárez, O., and Medina-Bañuelos, V. (2016). P300 detection based on eeg shape features. *Computational and mathematical methods in medicine*, 2016.

[Aricò et al., 2014] Aricò, P., Aloise, F., Schettini, F., Salinari, S., Mattia, D., and Cincotti, F. (2014). Influence of p300 latency jitter on event related potential-based brain–computer interface performance. *Journal of neural engineering*, 11(3):035008.

[Berndt and Clifford, 1994] Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA.

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167.

[Carabalona, 2017] Carabalona, R. (2017). The role of the interplay between stimulus type and timing in explaining bci-illiteracy for visual p300-based brain-computer interfaces. *Frontiers in neuroscience*, 11:363.

[Casarotto et al., 2005] Casarotto, S., Bianchi, A. M., Cerutti, S., and Chiarenza, G. A. (2005). Dynamic time warping in the analysis of event-related potentials. *IEEE Engineering in Medicine and Biology Magazine*, 24(1):68–77.

[Cecotti, 2015] Cecotti, H. (2015). Toward shift invariant detection of event-related potentials in non-invasive brain-computer interface. *Pattern Recognition Letters*, 66:127–134.

[Chakraborty and Horie, 2016] Chakraborty, G. and Horie, S. (2016). Towards an efficient and convenient brain computer interface. In *American Association of Artificial Intelligence Spring Symposium, Palo Alto, California, USA*, pages 21–23.

[Combaz and Van Hulle, 2015] Combaz, A. and Van Hulle, M. M. (2015). Simultaneous detection of p300 and steady-state visually evoked potentials for hybrid brain-computer interface. *PloS one*, 10(3):e0121481.

[Delaye and Anquetil, 2013] Delaye, A. and Anquetil, E. (2013). Hbf49 feature set: A first unified baseline for online symbol recognition. *Pattern Recognition*, 46(1):117–130.

[Draper and Smith, 1981] Draper, N. R. and Smith, Harry, .-j. a. (1981). *Applied regression analysis*. New York : Wiley, 2d ed edition. Includes index.

[Edelsbrunner et al., 1983] Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4):551–559.

[Farwell and Donchin, 1988] Farwell, L. A. and Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523.

[Fazel-Rezai et al., 2012] Fazel-Rezai, R., Allison, B. Z., Guger, C., Sellers, E. W., Kleih, S. C., and Kübler, A. (2012). P300 brain computer interface: current challenges and emerging trends. *Frontiers in neuroengineering*, 5.

[Fonseca and Jorge, 2000] Fonseca, M. J. and Jorge, J. A. (2000). Using fuzzy logic to recognize geometric shapes interactively. In *Proceedings of the 9th International Conference on Fuzzy Systems (FUZZ-IEEE'00)*, volume 1, pages 291–296, San Antonio, USA.

[Izenman, 2008] Izenman, A. J. (2008). *Modern multivariate statistical techniques*, volume 1. Springer.

[Kantardzic, 2011] Kantardzic, M. (2011). *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons.

[Kaper et al., 2004] Kaper, M., Meinicke, P., Grossekathoefer, U., Lingner, T., and Ritter, H. (2004). Bci competition 2003-data set iib: support vector machines for the p300 speller paradigm. *IEEE Transactions on Biomedical Engineering*, 51(6):1073–1076.

[Krusienski et al., 2006] Krusienski, D. J., Sellers, E. W., Cabestaing, F., Bayoudh, S., McFarland, D. J., Vaughan, T. M., and Wolpaw, J. R. (2006). A comparison of classification techniques for the p300 speller. *Journal of neural engineering*, 3(4):299.

[Krusienski et al., 2008] Krusienski, D. J., Sellers, E. W., McFarland, D. J., Vaughan, T. M., and Wolpaw, J. R. (2008). Toward enhanced p300 speller performance. *Journal of neuroscience methods*, 167(1):15–21.

[Liang and Bougrain, 2008] Liang, N. and Bougrain, L. (2008). Averaging techniques for single-trial analysis of oddball event-related potentials. In *4th International Brain-Computer Interface workshop*.

[Ma and Qiu, 2017] Ma, Z. and Qiu, T. (2017). Performance improvement of erp-based brain–computer interface via varied geometric patterns. *Medical & Biological Engineering & Computing*, pages 1–12.

[Manyakov et al., 2011] Manyakov, N. V., Chumerin, N., Combaz, A., and Van Hulle, M. M. (2011). Comparison of classification methods for p300 brain-computer interface on disabled subjects. *Computational intelligence and neuroscience*, 2011:2.

[Mirghasemi et al., 2006] Mirghasemi, H., Fazel-Rezai, R., and Shamsollahi, M. (2006). Analysis of p300 classifiers in brain computer interface speller. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 6205–6208. IEEE.

[Polich, 2007] Polich, J. (2007). Updating p300: an integrative theory of p3a and p3b. *Clinical neurophysiology*, 118(10):2128–2148.

[Polich and Heine, 1996] Polich, J. and Heine, M. R. (1996). P300 topography and modality effects from a single-stimulus paradigm. *Psychophysiology*, 33(6):747–752.

[Riccio et al., 2013] Riccio, A., Simione, L., Schettini, F., Pizzimenti, A., Inghilleri, M., Belardinelli, M. O., Mattia, D., and Cincotti, F. (2013). Attention and p300-based bci performance in people with amyotrophic lateral sclerosis. *Frontiers in human neuroscience*, 7.

[Sansana, 2016] Sansana, M. (2016). BCI-Based Spatial Navigation Control: A Comparison Study. Master's thesis, Faculdade de Ciências da Universidade de Lisboa.

[Selim et al., 2009] Selim, A. E., Wahed, M. A., and Kadah, Y. M. (2009). Machine learning methodologies in p300 speller brain-computer interface systems. In *Radio Science Conference, 2009. NRSC 2009. National*, pages 1–9. IEEE.

[Speier et al., 2014] Speier, W., Arnold, C., Lu, J., Deshpande, A., and Pouratian, N. (2014). Integrating language information with a hidden markov model to improve communication rate in the p300 speller. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(3):678–684.

[Speier et al., 2017] Speier, W., Deshpande, A., Cui, L., Chandravadia, N., Roberts, D., and Pouratian, N. (2017). A comparison of stimulus types in online classification of the p300 speller using language models. *PloS one*, 12(4):e0175382.

[Thulasidas et al., 2006] Thulasidas, M., Guan, C., and Wu, J. (2006). Robust classification of eeg signal for brain-computer interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(1):24–29.

[Townsend et al., 2010] Townsend, G., LaPallo, B., Boulay, C., Krusienski, D., Frye, G., Hauser, C., Schwartz, N., Vaughan, T., Wolpaw, J. R., and Sellers, E. (2010). A novel p300-based brain–computer interface stimulus presentation paradigm: moving beyond rows and columns. *Clinical Neurophysiology*, 121(7):1109–1120.

[Treder et al., 2011] Treder, M. S., Schmidt, N. M., and Blankertz, B. (2011). Gaze-independent brain–computer interfaces based on covert attention and feature attention. *Journal of neural engineering*, 8(6):066003.

[Turnip et al., 2017] Turnip, A., Amri, M. F., Fakrurroja, H., Simbolon, A. I., Suhendra, M. A., and Kusumandari, D. E. (2017). Deception detection of eeg-p300 component classified by svm method. In *Proceedings of the 6th International Conference on Software and Computer Applications*, pages 299–303. ACM.

[Vapnik and Vapnik, 1998] Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

[Vareka and Mautner, 2015] Vareka, L. and Mautner, P. (2015). Using the windowed means paradigm for single trial p300 detection. In *Telecommunications and Signal Processing (TSP), 2015 38th International Conference on*, pages 1–4. IEEE.

[Vieira, 2014] Vieira, J. (2014). mcali: Reconhecedor de esboços multiuso.

[Williamson et al., 2009] Williamson, J., Murray-Smith, R., Blankertz, B., Krauledat, M., and Müller, K.-R. (2009). Designing for uncertain, asymmetric control: Interaction design for brain–computer interfaces. *International Journal of Human-Computer Studies*, 67(10):827–841.

[Witten et al., 2016] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[Wronka et al., 2008] Wronka, E., Kaiser, J., and Coenen, A. M. (2008). The auditory p3 from passive and active three-stimulus oddball paradigm.

[Yin et al., 2015] Yin, E., Zeyl, T., Saab, R., Chau, T., Hu, D., and Zhou, Z. (2015). A hybrid brain–computer interface based on the fusion of p300 and ssvep scores. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(4):693–701.

[Zeyl et al., 2016] Zeyl, T., Yin, E., Keightley, M., and Chau, T. (2016). Adding real-time bayesian ranks to error-related potential scores improves error detection and auto-correction in a p300 speller. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(1):46–56.

[Zhang et al., 2016] Zhang, R., Li, Y., Yan, Y., Zhang, H., Wu, S., Yu, T., and Gu, Z. (2016). Control of a wheelchair in an indoor environment based on a brain–computer interface and automated navigation. *IEEE transactions on neural systems and rehabilitation engineering*, 24(1):128–139.